高等职业院校教学改革创新示范教材·软件开发系列

Android移动应用开发任务驱动教程

（Android Studio+Genymotion）

陈承欢　赵志茹　编著



電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书选择了目前开发Android应用程序的最佳组合：Android 5.0+Android Studio+Genymotion，使程序开发环境最佳、程序运行速度最快。

本书以Android应用程序的开发环境搭建、界面设计、功能实现、典型应用为主线选取教学内容和设置教学单元，将Android移动应用开发分为3个学习阶段（搭建与熟悉开发环境，界面设计和功能实现，Android的典型应用）和9个教学单元，每个教学单元面向教学全过程设置“教学导航→知识导读→任务实战→单元小结→单元习题”5个教学环节。每项任务设置了“任务描述→知识索引→实施过程”3个环节，通过“知识索引”将各项任务所应用的知识与“知识导读”环节的理论知识关联起来。采用“任务驱动、精讲多练、理论实践一体化”的教学方法，在完成每一项任务的过程中学习知识、训练技能、积累经验与固化能力。

本书适合作为高等院校计算机相关专业Android移动应用开发教材，也可作为Android程序设计的培训教材或参考书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

Android 移动应用开发任务驱动教程：Android Studio+Genymotion/陈承欢，赵志茹编著. —北京：电子工业出版社，2016.1
高等职业院校教学改革创新示范教材·软件开发系列
ISBN 978-7-121-27571-5

Ⅰ. ①A… Ⅱ. ①陈… ②赵… Ⅲ. ①移动终端－应用程序－程序设计－高等职业教育－教材
Ⅳ. TN929.53

中国版本图书馆CIP数据核字（2015）第271381号

策划编辑：程超群
责任编辑：郝黎明
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：19　字数：486.4 千字
版　　次：2016 年 1 月第 1 版
印　　次：2016 年 1 月第 1 次印刷
定　　价：39.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

PREFACE 前言

随着信息技术和通信技术的飞速发展，智能手机、平板电脑等移动智能终端已成为人们日常通信和信息处理的必备工具，并正在改变着人们的交流和生活方式。目前，Android 正成为移动终端的主流操作系统。应用 Android 不仅可以开发在智能移动设备上运行的应用软件，而且可以开发 2D 甚至 3D 游戏。高等学校也开始开设移动应用开发专业和 Android 移动应用开发课程。

Android 是一种基于 Linux 的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，由 Google 公司和开放手机联盟领导及开发，中文名称习惯称为“安卓”。

目前开发 Android 应用程序的最佳搭档是 Android 5.0+Android Studio+Genymotion。Android 5.0 是 Google 公司于 2014 年 10 月 15 日发布的全新 Android 操作系统。

Android Studio 是由 Google 公司推出的一个 Android 程序的开发环境，基于 IntelliJ IDEA，类似 Eclipse ADT，Android Studio 提供了集成的 Android 程序开发工具用于开发和调试。Android Studio 是一个全新的 Android 开发环境，成功解决了多分辨率、多语言等诸多程序开发与运行问题，开发者可以在编写程序的同时预览在不同尺寸屏幕中的外观效果。2015 年 5 月 29 日，Google 公司发布了 Android Studio 1.3 版，Android Studio 1.3 版使程序代码编写变得更加容易、快捷，而且支持 C++编辑和查错功能。

Genymotion 是一套完整的工具，它提供了 Android 虚拟环境。Genymotion 支持 Windows、Linux 和 Mac OS 等操作系统，由于其模拟运行速度快、画面流畅、容易安装和使用，且支持 OpenGL 3D 加速、多开、全屏等特性，并提供 Eclipse、IntelliJ IDEA 的开发插件，不仅是开发者们也是普通玩家们测试和运行 Android 软件和游戏的利器。

本书具有以下特色和创新：

（1）充分调研 Android 移动应用开发岗位的实际需求，精选教学案例

本书编写前期对 Android 应用程序的典型应用和 Android 移动应用开发职业岗位的需求做了大量细致的调研工作，先后分析了 300 多个 Android 典型应用程序的功能及实现方法，调研了 200 多个 Android 开发岗位的工作职责和职位要求。经过 4 次筛选、优化和简化，最终形成了 48 项典型教学案例。

（2）选择了最新、最佳开发工具组合 Android 5.0+Android Studio+Genymotion

经过反复多次调研和测试，本书选择了目前开发 Android 应用程序的最佳组合：Android

5.0+Android Studio+Genymotion，使程序开发环境最佳、程序运行速度最快，使所掌握的开发技术不过时。

（3）合理选取教学内容，科学设置教学单元

本书以 Android 应用程序的开发环境搭建、界面设计、功能实现、典型应用为主线选取教学内容和设置教学单元，而不是罗列 Android 相关知识。同时遵循学习者的认知规律和技能形成规律，将 Android 移动应用开发分为 3 个学习阶段：第 1 个阶段的重点是学会搭建与熟悉开发环境，第 2 个阶段的重点是学会界面设计和功能实现，第 3 个阶段的重点是学会 Android 的典型应用。本书设置了 9 个教学单元：Android 开发环境搭建与模拟器创建→Android Studio 的组成结构与基本操作→Android 的界面设计与资源应用程序设计→Android 的事件处理与交互实现程序设计→Android 的数据存储与数据共享程序设计→Android 的服务与广播应用程序设计→Android 的网络与通信应用程序设计→Android 的图像浏览与图形绘制程序设计→Android 的音频与视频应用程序设计。将 Android 应用开发的相关知识合理安排到各个教学单元。由于 Android 应用开发涉及众多的概念、理论和方法，同时限于本书篇幅的限制，本书重点探析了 Android 的基础知识和典型应用，主要介绍常用控件的应用。Spinner 控件、ListView 控件、自定义控件、Option Menu、Adapter、定时器、传感器、定位服务、动画处理技术等知识和技术没有过多涉及，请学习者参考相关书籍或网站进行深入学习。

（4）充分考虑教学实施的需求，合理设置教学流程

每个教学单元面向教学全过程设置“教学导航→知识导读→任务实战→单元小结→单元习题”5 个教学环节。每项任务设置了“任务描述→知识索引→实施过程”3 个环节，通过“知识索引”将各项任务所应用的知识与“知识导读”环节的理论知识关联起来，这样做既保证了 Android 应用开发相关理论知识的系统性和条理性，也凸现了知识的实际应用。

（5）采用“任务驱动、精讲多练、理论实践一体化”的教学方法，在完成每一项任务的过程中学习知识、训练技能、积累经验与固化能力

全书围绕 48 项 Android 移动开发任务，全方向促进 Android 应用开发能力的提升，引导学习者在完成各项设计任务的过程中，逐步理解概念和方法，循序渐进地学会开发环境搭建、界面设计和功能实现，从而熟练掌握形式多样的典型应用的实现方法和开发技巧。

本书由湖南铁道职业技术学院陈承欢教授和包头轻工职业技术学院赵志茹老师编著，包头轻工职业技术学院的张尼奇、广东科学技术职业学院的陈华政、四川交通职业技术学院的刘洋、长沙职业技术学院的殷正坤和艾娟、南通理工学院的卫丽华、湖南工业职业技术学院的刘曼春、宿州职业技术学院的胡秀建、长沙环保职业技术学院的杨茜、湖南铁道职业技术学院的颜谦和、冯向科、林东升、宁云智、肖素华、林保康、张丽芳等老师参与了教学案例的设计与部分章节的编写、校对、整理工作。

由于编者水平有限，教材中的疏漏之处敬请专家与读者批评指正，编者的 QQ 为 1574819688，本书免费提供电子教案、源代码等相关教学资源。

编　者

2016 年 1 月

CONTENTS 目录

单元 1　Android 开发环境搭建与模拟器创建 …… 1
【教学导航】 …… 1
【知识导读】 …… 1
1.1　基本概念 …… 1
1.2　Android 的体系结构 …… 3
1.3　Android Studio 的优势 …… 5
1.4　Genymotion 模拟器的优势 …… 6
1.5　在 BIOS 中设置处理器打开英特尔虚拟化技术（VT-x） …… 6
1.6　正确设置 JDK 或者 Android SDK 路径 …… 6
【任务实战】 …… 7
【任务 1-1】　搭建 Android 开发环境与创建模拟器 …… 7
【任务 1-2】　启动 Android Studio 与创建运行 Android Studio 项目 …… 9
【单元小结】 …… 15
【单元习题】 …… 15
单元 2　Android Studio 的组成结构与基本操作 …… 17
【教学导航】 …… 17
【知识导读】 …… 17
2.1　关于 Android 应用程序中的布局文件 activity_main.xml …… 17
2.2　关于 MainActivity.java 文件 …… 19
2.3　关于 AndroidManifest.xml 文件 …… 20
2.4　Android 应用程序的样式和主题设置 …… 23
2.5　关于 Android 系统的包 …… 24
2.6　相关问题剖析 …… 25
【任务实战】 …… 27
【任务 2-1】　熟悉 Android Studio 的组成结构 …… 27
【任务 2-2】　Android Studio 项目中的模块操作 …… 32
【任务 2-3】　Android Studio 开发环境的个性化设置 …… 34
【任务 2-4】　将 Android Studio 项目打包生成 APK …… 40
【单元小结】 …… 42
【单元习题】 …… 42
单元 3　Android 的界面设计与资源应用程序设计 …… 44
【教学导航】 …… 44
【知识导读】 …… 44
3.1　Android 屏幕元素的层次结构 …… 44
3.2　View 与 ViewGroup …… 46
3.3　Android 的布局对象 …… 47

3.4 Android 中创建 UI 界面的方式……50
3.5 Android 常用 UI 控件简介……54
3.6 Android 尺寸的单位……60
3.7 Android 控件的基本属性……60
3.8 TextView 控件与 EditText 控件……61
3.9 Button 控件……66
3.10 DatePicker 控件与 TimePicker 控件……66
3.11 Android 资源应用……68
【任务实战】……70
【任务 3-1】 使用文字标签显示欢迎信息……70
【任务 3-2】 设计包含多种控件的用户登录界面……75
【任务 3-3】 设计开关与调节声音的界面……78
【任务 3-4】 使用 DatePicker 控件获取日期……80
【任务 3-5】 使用日期时间控件获取与显示时间……81
【任务 3-6】 使用 LinearLayout 布局设计用户注册界面……82
【任务 3-7】 使用 TableLayout 布局设计商品信息展示界面……83
【任务 3-8】 使用 GridLayout 布局设计仿 Windows 8 桌面……86
【任务 3-9】 使用 FrameLayout 布局实现图片相框效果……88
【单元小结】……89
【单元习题】……90

单元 4 Android 的事件处理与交互实现程序设计……92
【教学导航】……92
【知识导读】……92
4.1 Android 的应用组件……92
4.2 Activity……93
4.3 Intent……100
4.4 Android 的事件处理机制……105
4.5 Android 的对话框与消息框……113
4.6 Android 输出日志信息的方法……116
4.7 OnTouchEvent……117
4.8 MotionEvent……117
【任务实战】……118
【任务 4-1】 用户触摸屏幕时显示提示信息……118
【任务 4-2】 用户登录时检测用户名的长度合法性……119
【任务 4-3】 获取屏幕单击位置……121
【任务 4-4】 实现动态添加联系人……123
【任务 4-5】 用户注册时检测 Email 格式……126
【任务 4-6】 打开浏览器浏览网页……128
【单元小结】……130
【单元习题】……130
单元 5 Android 的数据存储与数据共享程序设计……132
【教学导航】……132
【知识导读】……132
5.1 Android 系统的数据存储方式简介……132
5.2 使用 SQLite 数据库存储数据……133
5.3 使用 SharedPreference 对象存储数据……134

5.4　使用 ContentProvider 存储数据 ……136
5.5　使用 File 对象存储数据 ……136
5.6　URI 及其组成 ……137
【任务实战】……137
【任务 5-1】　设计可记住用户名和密码的登录界面 ……137
【任务 5-2】　对 Android 模拟器中的 SD 卡进行操作 ……144
【任务 5-3】　使用 SharedPreferences 实现 Activity 之间的数据传递 ……145
【任务 5-4】　使用 SQLite 数据库保存用户输入的用户名和密码 ……149
【任务 5-5】　预览选择的系统图片 ……154
【任务 5-6】　实现添加与查询联系人 ……157
【单元小结】……160
【单元习题】……160
单元 6　Android 的服务与广播应用程序设计 ……163
【教学导航】……163
【知识导读】……163
6.1　Service（服务） ……163
6.2　BroadcastReceiver（广播） ……173
【任务实战】……179
【任务 6-1】　获取系统的唤醒服务 ……179
【任务 6-2】　获取系统的屏蔽状态 ……182
【任务 6-3】　获取当前网络状态 ……184
【任务 6-4】　实现音量控制 ……187
【任务 6-5】　实现程序开机自动启动 ……190
【任务 6-6】　监控手机电池电量 ……193
【单元小结】……196
【单元习题】……196
单元 7　Android 的网络与通信应用程序设计 ……198
【教学导航】……198
【知识导读】……198
7.1　HTTP 协议简介 ……198
7.2　URL 请求的类别 ……199
7.3　Android 的线程与 Hander 消息机制 ……199
7.4　使用 HttpURLConnection 访问网络 ……202
7.5　使用 HttpClient 访问网络 ……203
【任务实战】……204
【任务 7-1】　获取指定城市的天气预报 ……204
【任务 7-2】　实现百度在线搜索 ……207
【任务 7-3】　实现浏览网络图片 ……211
【任务 7-4】　实现短信发送 ……214
【任务 7-5】　实现电话拨打 ……217
【单元小结】……219
【单元习题】……220
单元 8　Android 的图像浏览与图形绘制程序设计 ……221
【教学导航】……221
【知识导读】……221
8.1　使用简单图片 ……221
8.2　位图的典型应用 ……223
8.3　绘图 ……224
【任务实战】……227
【任务 8-1】　设计按钮切换的图片浏览器 ……227

【任务 8-2】 设计滑动切换的图片浏览器 ……229
【任务 8-3】 绘制简单几何图形 ……232
【任务 8-4】 绘制多种形式的路径 ……234
【任务 8-5】 绘制 Android 机器人图形 ……236
【单元小结】 ……238
【单元习题】 ……238
单元 9 Android 的音频与视频应用程序设计 ……240
【教学导航】 ……240
【知识导读】 ……240
9.1 SoundPool 类与播放音频 ……240
9.2 MediaPlayer 类与播放音频 ……242
9.3 VideoView 类与播放视频 ……246
9.4 MediaPlayer 与 SurfaceView 联合播放视频 ……248
【任务实战】 ……250
【任务 9-1】 使用 SoundPool 类播放音频 ……250
【任务 9-2】 使用 MediaPlayer 类播放本地音频 ……252
【任务 9-3】 实现在线音乐播放 ……257
【任务 9-4】 使用 VideoView 控件播放本地视频 ……262
【任务 9-5】 使用 MediaPlayer 类和 SurfaceView 控件播放本地视频 ……265
【单元小结】 ……269
【单元习题】 ……269
附录 A Android 移动应用开发课程设计 ……271
附录 B Android 移动应用开发岗位需求调研 ……273
附录 C 各单元任务中类及引入包的说明 ……275
附录 D 下载、安装与配置 JDK ……278
附录 E 下载与安装 Android Studio ……283
附录 F 下载与安装英特尔硬件加速执行管理器 ……286
附录 G 下载、安装与配置模拟器 Genymotion ……288
参考文献 ……296

单元 1

Android 开发环境搭建与模拟器创建

Android 是 Google 公司推出的移动设备开发平台，从 2007 年 11 月 5 日发布以来，短短几年的时间便成为全球最受欢迎的智能手机平台。应用 Android 不仅可以开发在手机或平板电脑等移动设备上运行的应用程序，而且可以开发 2D（2 Dimensions，平面游戏）甚至 3D（3 Dimensions，三维游戏）游戏。Android 应用开发是软件、硬件协同设计的一个过程，在开发 Android 应用程序之前，首先需要搭建一个方便、适用和高效的开发环境。一个性能良好、运行快捷的开发环境，可以使开发工作事半功倍。



教学导航

教学目标	（1）理解 Android、Android Studio、Genymotion 和 Gradle 等几个基本概念 （2）熟悉 Android 的系统架构的基本组成及其主要功能 （3）理解 Android Studio 和 Genymotion 模拟器的优势 （4）学会下载、安装与配置 JDK （5）学会下载与安装 Android Studio （6）学会下载、安装与配置模拟器 Genymotion （7）学会启动 Android Studio 以及创建、运行 Android Studio 项目
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	4 课时



知识导读

1.1 基本概念

1．Android

Android 一词的本义指“机器人”，同时也是 Google 于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，该平台由操作系统、中间件、用户界面和应用软件组成。Android 是一种基于 Linux 的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，由 Google 公司和开放手机联盟领导及开发。Android 操作系统最初由 Andy Rubin 开发，主要支持手机，后来 Android 逐渐扩展到平板电脑及其他领域上，如电视、数码相机、游戏机等。

Android 是一个完全开放的操作系统，开放的平台允许任何移动终端厂商加入到 Android

联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。由于 Android 的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。Android 应用程序具有系统资源占用少、系统运行速度快、个性化的操作界面、操作简单、有众多的软件提供给消费者下载等诸多优势。

2．Android Studio

Android Studio 是 Google 开发的一款面向 Android 开发者的 IDE（Integrated Development Environment，集成开发环境），支持 Windows、Mac、Linux 等操作系统，基于 Java 语言集成开发环境 IntelliJ 搭建而成。此款 IDE 在 2013 年 5 月的 Google I/O 开发者大会上首次露面，之后推出了若干个测试版，直到 2014 年 12 月 8 日，Google 终于正式发布了面向 Android 开发者的集成开发环境 Android Studio 1.0 稳定版。2015 年 5 月 29 日，在 Google 的 I/O 开发者大会上，发布了 Android Studio 1.3 版，Android Studio 1.3 版使代码编写变得更加容易，速度得到提升，而且支持 C++编辑和查错功能。

Android Studio 是一项全新的基于 IntelliJ IDEA 的 Android 开发环境，类似于 Eclipse ADT 插件，Android Studio 提供了集成的 Android 开发工具用于开发和调试。Android Studio 是一款性能良好的 Android 应用开发工具，其主要特点如下：

① 基于 Gradle 的构建支持。

② Android 特定重构和快速修复。

③ 提示工具更好地对程序性能、可用性、版本兼容和其他问题进行控制捕捉。

④ 支持 ProGuard 和应用签名功能。

⑤ 基于模板的向导来生成常用的 Android 应用设计和控件。

⑥ 功能强大的布局编辑器，支持拖放 UI 控件并进行效果预览。

3．Genymotion

Genymotion 是一款高性能的 Android 模拟器，它支持 Windows、Mac、Linux，由于其模拟运行速度很快、画面流畅、且支持 OpenGL 3D 加速、多开、全屏等特性，并提供 Eclipse、IntelliJ IDEA（Java 语言开发的集成环境）的开发插件，不仅是开发者也是普通玩家们测试和运行 Android 软件和游戏的利器。

4．Gradle

Gradle 是以 Groovy 语言为基础，面向 Java 应用为主。它抛弃了基于 XML 的各种繁琐配置，是基于 DSL（领域特定语言）语法的自动化构建工具。Gradle 可以用于 Android 开发的新一代的 Build System，也是 Android Studio 默认的 build 工具。因为 Groovy 是 JVM 语言（Groovy），所以可以使用大部分的 Java 语言库。所谓 DSL 就是专门针对 Android 开发的插件，例如标准 Gradle 之外的一些新的方法（Method）、闭包（Closure）等。由于 Gradle 的语法足够简洁，而且可以使用大部分的 Java 包，当之无愧地成为新一代 Build System。

使用 Android Studio 新建一个项目后，默认会生成两个 build.gralde 文件，一个位于项目根目录，一个位于 app 目录下。还有另外一个文件 settings.gradle。根目录下的脚本文件是针对 module 的全局配置，它的作用域所包含的所有 module 是通过 settings.gradle 来配置的。app 文件夹就是一个 module，如果在当前项目中添加了一个新的 module ，就需要在 settings.gralde 文件中包含这个新的 module。

5．APK

APK（Android Package 的缩写）是 Android 应用程序的安装包（apk），类似 Symbian Sis

或 Sisx 的文件格式。通过将 APK 文件直接传到 Android 模拟器或 Android 手机中执行即可安装。apk 文件和 sis 一样，把 android sdk 编译的项目打包成一个安装程序文件，格式为 apk。APK 文件其实是 zip 格式，但后缀名被修改为 apk，通过 UnZip 解压后，可以看到 Dex 文件，Dex 是 Dalvik VM executes 的全称，即 Android Dalvik 执行程序，并非 Java ME 的字节码，而是 Dalvik 字节码。

1.2 Android 的体系结构

英文版 Android 体系结构如图 1-1 所示，中文版 Android 体系结构如图 1-2 所示。

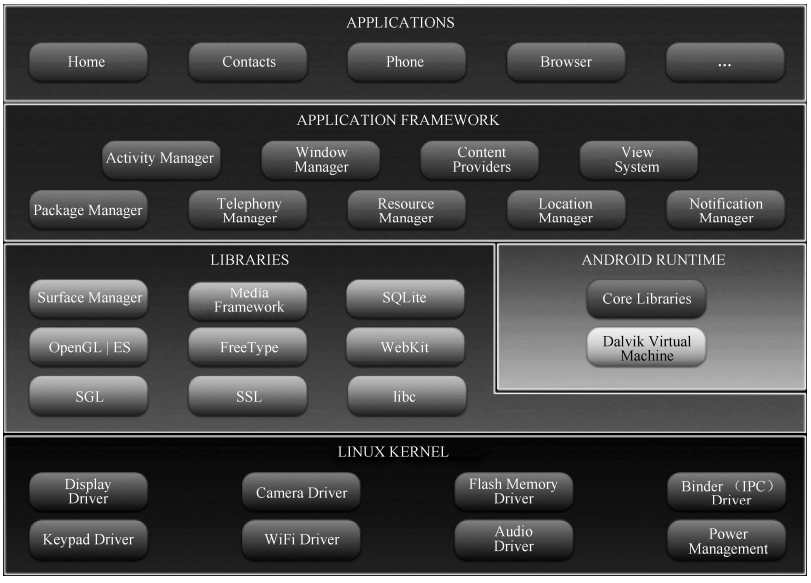


图 1-1　Android 体系结构（英文版）



图 1-2　Android 体系结构（中文版）

由图1-1和图1-2可以很明显看出，Android体系结构由5部分组成，分别是：Linux Kernel、Android Runtime、Libraries、Application Framework、Applications。

（1）Linux Kernel

Android 基于 Linux 2.6 提供核心系统服务，例如安全管理、内存管理、进程管理、网络堆栈、驱动模型等。Linux Kernel 也作为硬件和软件之间的抽象层，它隐藏具体硬件细节而为上层提供统一的服务。

（2）Android Runtime

Android 包含一个核心库的集合，提供大部分在 Java 编程语言核心类库中可用的功能。每一个 Android 应用程序是 Dalvik 虚拟器中的实例，运行在它们自己的进程中。Dalvik 虚拟器设计成在一个设备可以高效地运行多个虚拟器。Dalvik 虚拟器可执行文件的格式是 dex，dex 格式是专为 Dalvik 设计的一种压缩格式，适合内存和处理器速度有限的系统。

大多数虚拟器包括 JVM 都是基于栈的，而 Dalvik 虚拟器则是基于寄存器的，Dalvik 虚拟器依赖于 Linux 内核提供基本功能，例如线程和底层内存管理。

（3）Libraries

Android 包含一个 C/C++库的集合，供 Android 系统的各个控件使用。这些功能通过 Android 的应用程序框架（Application Framework）暴露给开发者。下面列出一些核心库：

① 系统 C 库：标准 C 系统库（libc）的 BSD 衍生，基于嵌入式 Linux 设备。

② 媒体库：基于 PacketVideo 的 OpenCORE。这些库支持播放和录制许多流行的音频和视频格式以及静态图像文件，包括 MPEG4、H.264、MP3、AAC、AMR、JPG、PNG 等。

③ 界面管理：管理访问显示子系统和无缝组合多个应用程序的二维和三维图形层。

④ LibWebCore：新式的 Web 浏览器引擎，驱动 Android 浏览器和内嵌的 Web 视图。

⑤ SGL：基本的 2D 图形引擎。

⑥ 3D 库：基于 OpenGL ES 1.0 APIs 的实现，库使用硬件 3D 加速或包含高度优化的 3D 软件光栅。

⑦ FreeType：位图和矢量字体渲染。

⑧ SQLite：所有应用程序都可以使用的强大而轻量级的关系数据库引擎。

（4）Application Framework

通过提供开放的开发平台，Android 使开发者能够编写极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知，等等。

开发者可以完全使用核心应用程序所使用的框架 APIs。应用程序的体系结构旨在简化控件的重用，任何应用程序都能发布它的功能且任何其他应用程序可以使用这些功能，但需要服从框架执行的安全限制。

所有的应用程序其实是一组服务和系统，主要包括：

① 视图（View）：丰富的、可扩展的视图集合，可用于构建一个应用程序，包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器。

② 内容提供者（Content Providers）：使应用程序能访问其他应用程序的数据，或共享自己的数据。

③ 资源管理器（Resource Manager）：提供访问非代码资源，如本地化字符串、图形和布

局文件等。

④ 通知管理器（Notification Manager）：使所有的应用程序能够在状态栏显示自定义警告。

⑤ 活动管理器（Activity Manager）：管理应用程序生命周期，提供通用的导航回退功能。

（5）Applications

Android 装配一个核心应用程序集合，包括电子邮件客户端、SMS 程序、日历、地图、浏览器、联系人和其他设置。所有应用程序都是用 Java 编程语言编写的。

由以上分析可知 Android 的架构是分层的、分工很明确。Android 本身是一套“软件堆叠（Software Stack）”，或称为“软件叠层架构”，叠层主要分成三层：操作系统、中间件和应用程序。

1.3 Android Studio 的优势

Android Studio 是 Google 推出，专门为 Android“量身订做”的，是 Google 大力支持的一款基于 IntelliJ IDEA 改造的 IDE，它具有以下优势：

（1）速度更快

Android Studio 在启动速度、响应速度、内存占用等方面都全面领先于 Eclipse。

（2）UI 更漂亮

Android Studio 自带的 Darcula 主题的炫酷黑界面比 Eclipse 下的黑色主题更漂亮。

（3）更加智能

Android Studio 更加智能，能够实现自动保存文档，从此再也不用每次编辑完代码都按【Ctrl+S】组合键去保存了，熟悉 Android Studio 以后，应用程序开发效率将会大大提升。能够实现自动注册 Activity，之前我们用 Eclipse 开发 Android 应用，添加一个新的 activity 后，都要到 Manifest.xml 里面注册，现在我们添加完新的 Activity 后，Android Studio 会自动为我们注册 Activity。

（4）整合了 Gradle 构建工具

Gradle 是一个新的构建工具，自 Android Studio 亮相之初就支持 Gradle，可以说 Gradle 集合了 Ant 和 Maven 的优点，不管是配置、编译、打包都非常棒。

（5）强大的 UI 编辑器

Android Studio 的编辑器非常智能，除了吸收 Eclipse+ADT 的优点之外，还自带了多设备的实时预览，这对 Android 开发者来说简直是神器。

（6）内置终端

Android Studio 内置终端，这对于习惯命令行操作的人来说简直是福音，再也不用来回切换了，一个 Android Studio 全部搞定。

（7）更完善的插件系统

Android Studio 下支持各种插件，如 Git、Markdown、Gradle 等，想要什么插件，可以直接搜索下载。

（8）完美整合版本控制系统

安装的时候就自带了如 GitHub、 Git、SVN 等流行的版本控制系统，可以直接迁出项目。

1.4 Genymotion 模拟器的优势

Genymotion 模拟器的主要优势如下：

（1）速度快性能好，资源占用低，系统要求提供 512MB 内存就能运行。

（2）支持 OpenGL 3D 加速，可以流畅玩大型 3D 游戏。

（3）支持同时启动多个模拟器，可以实现软件或游戏开发。

（4）支持多种虚拟传感器，如电池状态、GPS、NFC、重力加速器等模拟，在微信之类的 App 上有特别用处。

（5）支持摄像头、蓝牙等设备。

（6）可以配置模拟器参数，如屏幕分辨率、内存大小、CPU 数量。

（7）支持多种分辨率，直接拖曳窗口即可调整画面大小。

（8）支持几乎全部版本的 Android 系统，从 2.x～5.x。

（9）支持直接拖放文件到虚拟设备。

（10）启动速度和运行速度都快于 Android SDK 官方的模拟器。

1.5 在 BIOS 中设置处理器打开英特尔虚拟化技术（VT-x）

按【F1】键进入 BIOS 设置实用程序，使用箭头键，选择“setup”，按【Enter】键，再选择“Advanced”，然后按【Enter】键，选择“Intel(R) Virtualization Technology”，按【Enter】键选择“Enabled”，然后按【Enter】键继续，按【F10】键以保存并退出，选择“Yes”，按【Enter】键，完全关机（关闭电源），等待几秒钟，重新启动计算机。

1.6 正确设置 JDK 或者 Android SDK 路径

有时运行 Android Studio 会提醒 Android SDK 或者 JDK 不存在，此时需要重新设置，设置方法如下：

在“Welcome to Android Studio”界面中，“Quick Start”选择“Configure”选项，切换到“Configure”界面，在该界面选择“Project Defaults”选项，如图 1-3 所示，切换到如图 1-4 所示的“Project Defaults”界面，在该界面选择“Project Structure”选项，打开“Project Structure”对话框，在该对话框中设置 Android SDK 和 JDK 的路径，如图 1-5 所示。

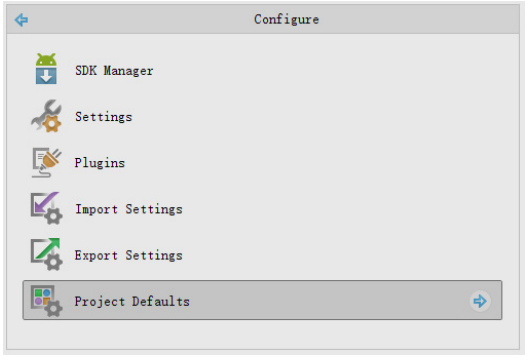


图 1-3 “Configure”界面

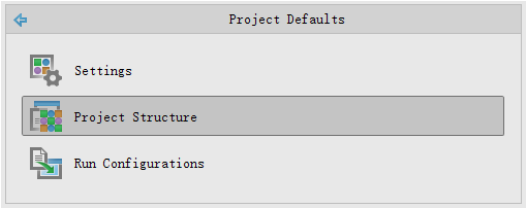


图 1-4 “Project Defaults”界面

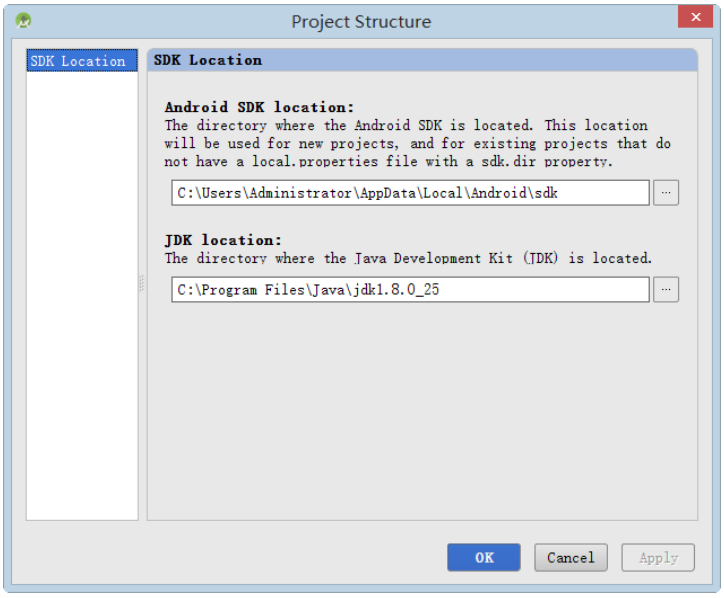


图 1-5 “Project Structure”对话框

在 Android Studio 中依次选择命令【File】→【Other Settings】→【Default Project Structure】，也可以打开“Project Structure”对话框。



“工欲善其事，必先利其器”，在学习 Android 开发之前，必须先熟悉并搭建其所需要的开发环境。本书以 Windows 7 操作系统为例，学习 Android 的开发。Android 开发相关的软件及其下载地址如表 1-1 所示。

表 1-1　Android 开发相关的软件及其下载地址

序号	软件名称	下载地址	本书使用的版本
1	JDK	http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html	Java SE Development Kit 8u45
2	Android SDK	http://www.android-studio.org/	Android Studio V1.2.1
3	英特尔硬件加速执行管理器	https://software.intel.com/zh-cn/android/articles/intel-hardware-accelerated-execution-manager/	haxm-windows_r05
4	Genymotion	http://www.genymotion.net/	genymotion-2.4.0-vbox.exe

【任务 1-1】搭建 Android 开发环境与创建模拟器

【任务描述】

1．下载、安装与配置 JDK

在开发 Android 应用程序之前，首先必须安装 JDK 控件，JDK 包括运行 Java 程序所必需的 JRE（Java Runtime Environment）以及开发过程中常用的库文件。

（1）首先从网上下载最新版本的 JDK。

（2）下载完成 JDK 的安装文件后，双击启动安装文件，然后只需要按照安装向导提示的步骤进行安装即可。

（3）JDK 安装完成后，需要对 JAVA_HOME 以及 Path、ClassPath 两个系统环境变量进行正确的配置。其中 JAVA_HOME 值设置为 JDK 所在路径，例如作者电脑为“C:\Program Files\Java\jdk1.8.0_25”，Path 环境变量为“bin”文件夹所在路径，例如作者电脑中 Path 变量设置为“C:\Program Files\Java\jdk1.8.0_25\bin;”，注意“;”是与其他路径之间的分隔符。ClassPath 环境变量设置编译 Java 程序时所需要的一些外部的 class 文件所在路径，例如作者电脑中 ClassPath 变量设置为“C:\Program Files\Java\jdk1.8.0_25\jre\lib;”。

2．下载与安装 Android Studio

从网上下载最新版本的 Android Studio，可以根据电脑中有没有 Android SDK 来选择下载是否包含 SDK 的安装文件，建议使用包含 SDK 的安装包（exe）。

3．下载与安装英特尔硬件加速执行管理器

英特尔®硬件加速执行管理器是一个硬件辅助的虚拟化引擎（Hypervisor，虚拟器监视器），它使用英特尔®虚拟化技术加速安卓应用程序在主机上的模拟。英特尔®硬件加速执行管理器与英特尔提供的安卓 x86 模拟器映像及官方安卓 SDK Manager（安卓软件开发套件）相结合，可在启用英特尔虚拟器的系统上更快地模拟安卓系统，这将大大提高启动速度和程序的运行速度，允许 Android 模拟器能够以原始速度（真机运行速度）运行在使用 Intel x86 处理器的电脑中。试下载与安装英特尔硬件加速执行管理器。

4．下载、安装与配置模拟器 Genymotion

安卓 SDK 自带的安卓模拟器启动速度有点慢，而 Genymotion 号称最快的安卓模拟器，当然这里的“快”指的是启动速度快和操作流畅。这个安卓模拟器是基于 Virtual Box 的，优点是速度快、设备类型多、功能丰富，不过要注册才能下载。试下载、安装与配置模拟器 Genymotion。

【知识索引】

（1）JDK 的概念与特点。

（2）JDK 的下载、安装与配置。

（3）Android Studio 的基本概念、使用特点与优势。

（4）Android Studio 的下载与安装。

（5）英特尔硬件加速执行管理器的性能与功用。

（6）英特尔硬件加速执行管理器的下载、安装与测试。

（7）Genymotion 功用与优势。

（8）Genymotion 的下载与安装。

（9）Android 与 Genymotion 的集成配置。

（10）Android 模拟器的添加与启动。

【实施过程】

1．下载、安装与配置 JDK

下载、安装与配置 JDK 的实施过程详见附录 D。

2．下载与安装 Android Studio

下载与安装 Android Studio 的实施过程详见附录 E。

3．下载与安装英特尔硬件加速执行管理器

下载与安装英特尔硬件加速执行管理器的实施过程详见附录 F。

4．下载、安装与配置模拟器 Genymotion

下载、安装与配置模拟器 Genymotion 的实施过程详见附录 G。

【任务 1-2】 启动 Android Studio 与创建运行 Android Studio 项目

【任务描述】

（1）安装 Android Studio 完成后，启动 Android Studio。

（2）在 Android Studio 启动过程中创建 Android Studio 项目 Application0101。

（3）在 Android Studio 主窗口中创建 Android Studio 项目 App0101。

（4）运行 Android Studio 项目 App0101。

【知识索引】

（1）Android Studio 的启动过程、集成环境的基本组成、菜单组成与工具栏。

（2）在 Android Studio 启动过程中创建 Android Studio 项目。

（3）在 Android Studio 主窗口中创建 Android Studio 项目。

（4）Android Studio 项目的运行方法。

【实施过程】

1．启动 Android Studio

安装 Android Studio 完成后，在 Windows 操作系统的桌面双击快捷方式“Android Studio”或者在“开始”菜单中选择“Android Studio”命令，即可启动 Android Studio，启动画面如图 1-6 所示。



图 1-6　Android Studio 的启动画面

安装完成后第一次启动 Android Studio，会显示【Complete Installation】对话框，该界面用以选择导入 Android Studio 的配置文件，有两个选项，第 1 个选项用于导入以前版本的配置文件，第 2 个不导入配置文件。如果本机以前曾安装使用过 Android Studio，可以选择以前的版本。如果是第一次安装使用，可以选择第 2 项，如图 1-7 所示。

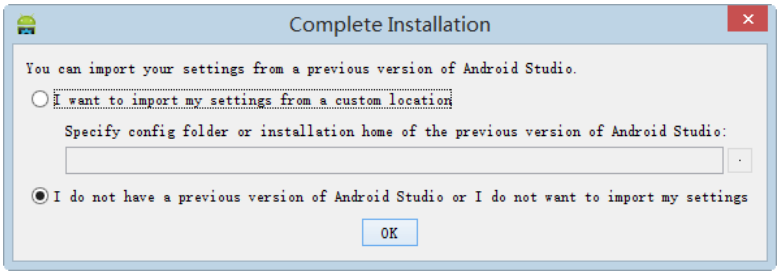


图 1-7 “Complete Installation”对话框

然后单击【OK】按钮，开始检查并获取 Android SDK 控件信息，如图 1-8 所示。

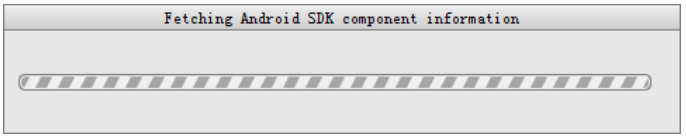


图 1-8 获取 Android SDK 控件信息

有时这一步会等待很长时间，很大的原因就是网络连接有问题。如果想跳过这一步，可以在 Android Studio 安装文件夹的 bin 子文件夹下，找到 idea.properties 文件，在文件最后添加 1 行代码：disable.android.first.run=true 。

如果需要检查更新，等待下载与安装更新需要一段时间，Android SDK 更新完毕，就可以看到如图 1-9 所示的“Welcome to Android Studio”界面。



图 1-9 “Welcome to Android Studio”界面

该界面左侧显示“Recent Projects”，由于第一次启动还没有创建项目，所以目前为空。右侧显示“Quick Start”，有多个选项，其功能如下：

选项 1：创建一个 Android Studio 项目。

选项 2：打开一个 Android Studio 项目。

选项 3：导入官方样例，会从网络上下载代码。

选项 4：从版本控制直接迁出，支持 CVS、SVN、Git、Mercurial，甚至 GitHub。

选项 5：导入非 Android Studio 项目，例如 Eclipse Android 项目、IDEA Android 项目。

选项 6：设置。

选项 7：帮助文档。

如果一些选项无法选择，说明本机的 Android SDK 或者 JDK 路径设置有问题，参考【知识导读】环节介绍的方法设置 Android SDK 或者 JDK 的路径。

2．在 Android Studio 启动过程中创建 Android Studio 项目 Application0101

接下来，我们开始创建第一个项目，单击【Start a new Android Studio project】，然后显示【Create New Project】之“New Project”界面。在“Application name（应用名称）”文本框中输入应用程序名称，它是 App 在设备上显示的应用程序名称，也是在 Android Studio Project 的名称，这里输入“Application0101”；在“Company Domain（公司域名）”文本框中输入公司的网址，这里保留默认值不变，这里的内容决定了项目的包名（Package name）；“Package name（应用程序包名）”，每一个 App 都有一个独立的包名，如果两个 App 的包名相同，Android 会认为它们是同一个 App。因此，需要尽量保证，不同的 App 拥有不同的包名；在“Project location（项目存放位置）”设置项目合适的保存位置。输入或设置完成后如图 1-10 所示。

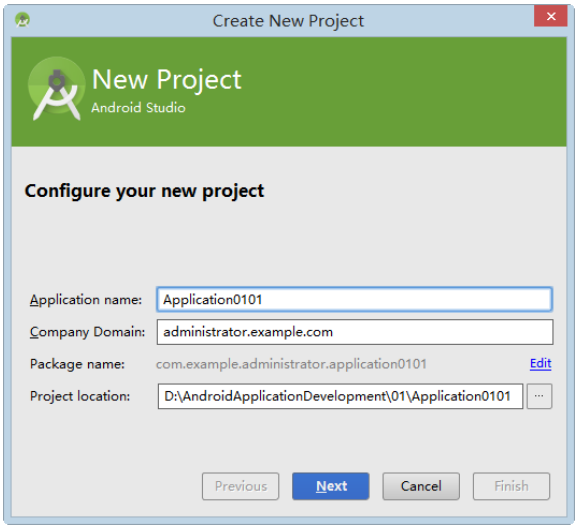


图 1-10　【Create New Project】之“New Project”界面

单击【Next】按钮，进入【Create New Project】之“Target Android Devices”界面，这里可以看到几个选项，默认的选择是“Phone and Tablet”（手机和平板），还可以选择“TV”、Wear 和 Glass 等。Minimum SDK 表示的是 Module 支持的 Android 最低版本，不同的用户可以选择不同的版本，这里在“Minimum SDK”列表框中选择“API 21:Android 5.0(Lollipop)，如图 1-11 所示。

单击【Next】按钮进入“Add an activity to Mobile”界面，设置 Android 应用程序启动时的运行界面，这里选择默认的 Blank Activity（空白界面），如图 1-12 所示，也可以根据自己的需要选择其他类型界面。

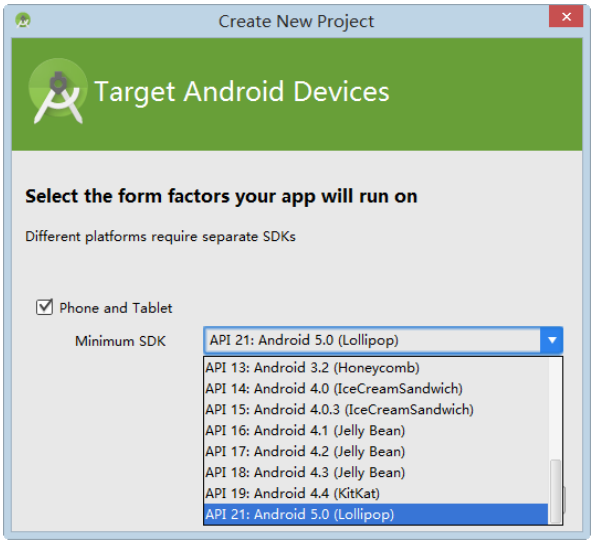


图 1-11 【Create New Project】之“Target Android Devices”界面

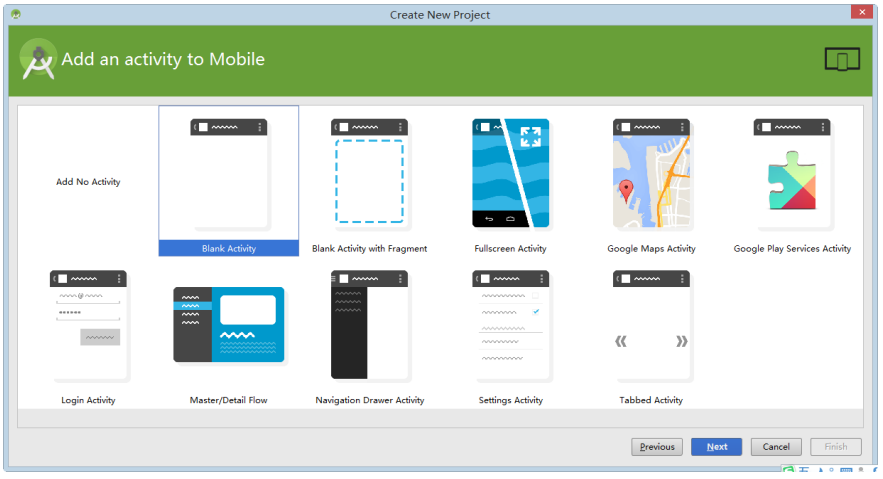


图 1-12 “Add an activity to Mobile”界面

单击【Next】按钮进入“Customize the Activity”界面，该界面让我们输入 Activity 的一些相关信息。在“Activity Name”文本框中输入 Activity 名称，在“Layout Name”文本框中输入布局名称，在“Title”文本框中输入 Activity 上显示的标题内容，在“Menu Resource Name”输入菜单栏的资源文件名称，这里可以使用默认值，如图 1-13 所示，也可以根据实际需要输入合适的内容。

单击【Finish】按钮后，等一会出现如图 1-14 所示的进度条，这里需要下载 Gradle，并且只有第一次会下载，有点慢，请耐心等待，Android Studio 会为我们打开刚才创建的新项目，至此一个简单的 Android Studio 项目就创建完成了，完整的项目界面如图 1-15 所示。

可以看到上面的窗口大致有从左到右四个区域（默认选中的是 activity_main.xml 用户界面配置文件），第 1 个区域是项目列表区（每个项目的代码和资源都在这里列出），第 2 个区域是所有可用控件，第 3 个区域是可视化的布局区（看到一个手机），第 4 个区域是控件的树形列表及属性设置。

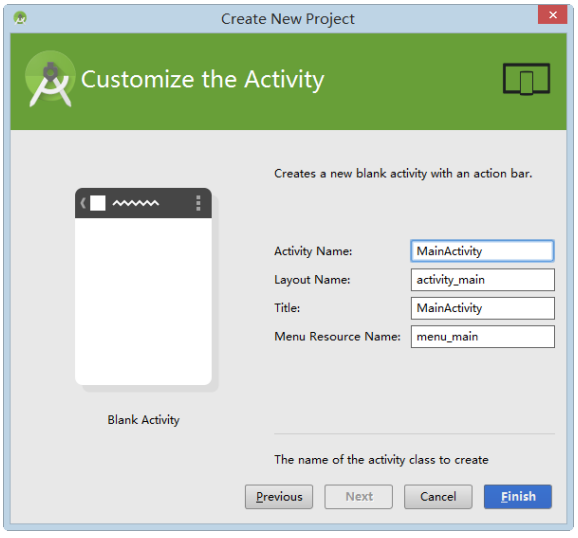


图 1-13 “Customize the Activity”界面

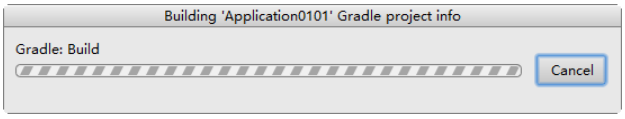


图 1-14 “Gradle Build”进度条

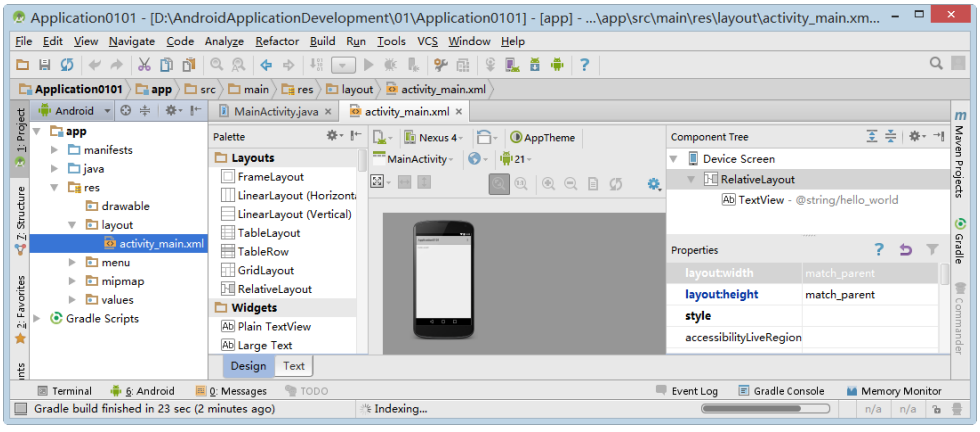


图 1-15　完整的 Android Studio 项目界面

3．在 Android Studio 主窗口中创建 Android Studio 项目 App0101

打开 Android Studio 主窗口，在【File】菜单中选择命令“New Project”，如图 1-16 所示。

显示【Create New Project】之“New Project”界面，在“Application name”文本框中输入“App0101”，在“Company Domain”文本框中输入“example.com”。

单击【Next】按钮进入下一步，创建项目 App0101，后面各个步骤与创建项目 Application0101 类似，最后单击【Finish】按钮，完成项目 App0101 的创建。

4．运行 Android Studio 项目 App0101

在 Android Studio 主窗口的工具栏中单击绿色箭头按钮▶，打开“Choose Device”对话框，在该对话框中选择“Choose a running device”单选按钮，然后在设备列表选择正在运行的

Android 模拟器，如图 1-17 所示。

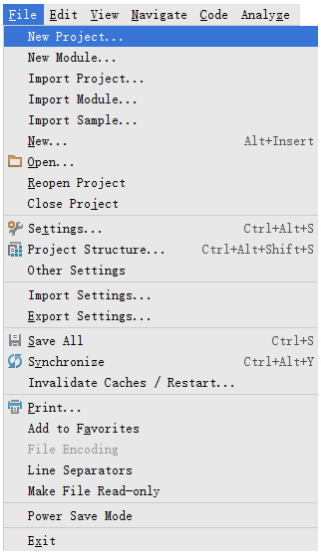


图 1-16　在【File】菜单中选择命令“New Project”

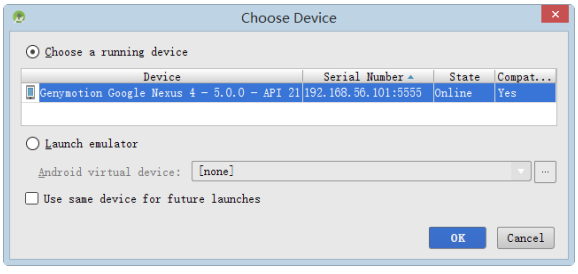


图 1-17 “Choose Device”对话框

然后单击【OK】按钮，Android 模拟器开始运行，应用程序 App0101 在模拟器中的运行结果如图 1-18 所示。

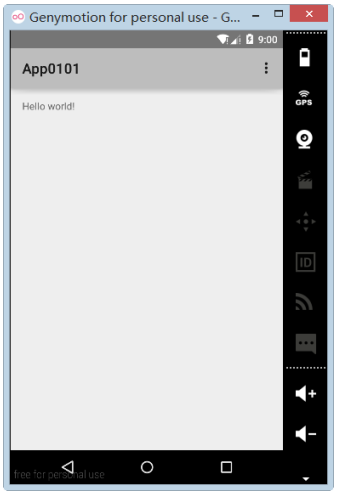


图 1-18　应用程序 App0101 在模拟器中的运行结果



单元小结

Android 是 Google 公司基于 Linux 平台开发的手机及平板电脑的系统，自问世以来，发展迅速、受到了前所未有的关注，也成为移动平台最受欢迎的操作系统之一。本单元主要介绍了 Android 的相关概念以及体系结构，通过完成 Android 程序开发与运行环境的搭建任务以及第 1 个简单 Android 程序的开发任务，我们便开始了 Android 应用程序开发之旅。



单元习题

一、填空题

1．Android 第 1 个版本 Android1.1 是（　　）年 9 月发布的。

2．Android 是 Google 公司基于（　　）平台开发的手机及平板电脑的（　　）。

3．Android 系统采用分层架构，由高至低分别为：（　　）、（　　）、（　　）和（　　）。

4．Android 的分层架构中，应用框架层使用（　　）语言开发，核心库使用（　　）语言开发。

5．程序员编写 Android 应用程序时，主要调用（　　）层提供的接口实现。

6．在 Android 智能终端中，有很多应用，如拍照、管理联系人等，它们都属于 Android 的（　　）层。

7．为了让程序员更加方便地运行调试程序，Android 提供了（　　），可以方便地运行程序，而不需要真实的移动终端。Genymotion 是一款优秀专业高性能的（　　）。

8．Android Studio 是一项全新的基于（　　）的 Android 开发环境，类似于 Eclipse ADT 插件，Android Studio 提供了集成的（　　）开发工具用于开发和调试。

二、选择题

1．创建程序的过程中，填写 Application Name 表示（　　）。

A．应用程序名称　　B．项目名称　　C．项目包名　　D．类名称

2．Android 操作系统的手机可以有以下几个方法进行软件安装（　　）。（多选题）

A．通过手机直接登录百度网站下载安装

B．通过手机直接登录 Android 门户网站进行下载安装

C．通过数据线与电脑连接直接下载安装

D．通过 PC 终端上网下载至 SD 卡里再插入手机进行安装

3．Android 操作系统主要的特点是（　　）。（多选题）

A．与 Google 无缝结合　　B．实现个性化应用

C．良好的平台开放性　　D．软件均要收费

4．以下对 Android 操作系统的描述中正确的是（　　）。（多选题）

A．系统资源占用少，系统运行速度快

B．完全开放性的操作系统，对所有软件开发商、手机厂商、运营商平等享有

C．有众多的软件提供给消费者下载

D．个性化的操作界面，操作简单

5．以下哪些是 Android Studio 的优势（　　）。（多选题）

A．启动速度、响应速度更快　　B．更加智能

C．整合了 Gradle 构建工具　　D．强大的 UI 编辑器

6．以下哪些是 Genymotion 模拟器的优势（　　）。（多选题）

A．支持同时启动多个模拟器　　B．支持多种虚拟传感器

C．支持多种分辨率　　D．支持 OpenGL 3D 加速

三、简答题

1．Android 的四层架构分别包括哪几层？分别起到什么作用？

2．简述 Android 开发环境安装的步骤。

3．简述 Android Studio 的优势。

4．简述 Genymotion 模拟器的优势。

5．简述 Android 应用程序创建和运行的步骤。

单元 2

Android Studio 的组成结构与基本操作

Android Studio 是一个 Android 开发环境，基于 IntelliJ IDEA 类似 Eclipse ADT，Android Studio 提供了集成的 IDE 用于开发和调试 Android 应用程序。Google（谷歌）已宣布 Android Studio 将取代 Eclipse，正式成为官方集成开发工具，并中止对 Eclipse ADT 插件的支持。本单元主要认识 Android Studio 的组成结构，完成项目的模块操作、开发环境的个性设置等基本操作。

教学导航

教学目标	（1）了解 Android 应用程序中的布局文件 activity_main.xml、MainActivity.java 文件和 AndroidManifest.xml 文件的基本组成及其含义 （2）认识 Android 应用程序的样式和主题设置 （3）熟悉 Android Studio 的组成结构 （4）学会 Android Studio 项目中的模块操作 （5）学会 Android Studio 开发环境的个性化设置 （6）学会将 Android Studio 项目打包生成 APK
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	6 课时

知识导读

2.1 关于 Android 应用程序中的布局文件 activity_main.xml

在 Android 应用程序中，界面是通过布局文件设定的，布局文件采用 XML 格式。每一个 Android 项目成功创建后，默认生成一个布局文件 activity_main.xml，该文件位于项目的 res/layout 文件夹中，其默认代码如表 2-1 所示，打开该布局文件可以看到一个布局文件窗口，如图 2-1 所示。

表 2-1　activity_main.xml 文件中默认生成的代码

序号	代　码
01	<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	xmlns:tools=http://schemas.android.com/tools
03	android:layout_width="match_parent"
04	android:layout_height="match_parent"

续表

序号	代　　码
05	android:paddingLeft="@dimen/activity_horizontal_margin"
06	android:paddingRight="@dimen/activity_horizontal_margin"
07	android:paddingTop="@dimen/activity_vertical_margin"
08	android:paddingBottom="@dimen/activity_vertical_margin"
09	tools:context=".MainActivity">
10	<TextView android:text="@string/hello_world"
11	android:layout_width="wrap_content"
12	android:layout_height="wrap_content" />
13	</RelativeLayout>

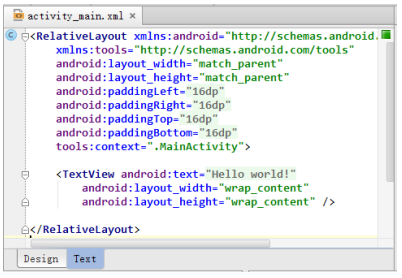


图 2-1　布局文件窗口

从图 2-1 可以看出该布局文件窗口有两个选项卡，分别是“Design”和“Text”，其中“Design”选项卡是布局文件的图形化视图，如图 2-2 所示。在该图形化视图中可以通过鼠标将 Palette 窗口中的控件直接拖动到界面中，让界面变得更加美观、友好。

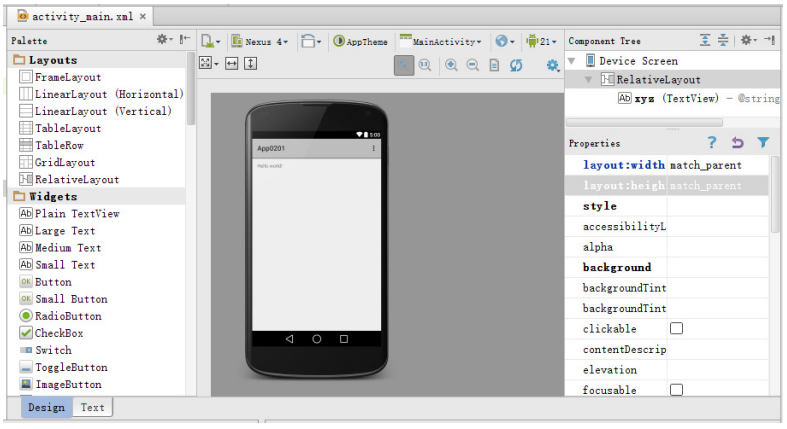


图 2-2　布局文件窗口的“Design”选项卡

从表 2-1 中布局文件 activity_main.xml 的代码可以看出，新建的 Android 程序默认的布局方式是相对布局（RelativeLayout），该布局中包含一个文本控件（TextView）。要让布局文件或者控件能够显示在界面上，必须要设置 RelativeLayout 和控件的宽度和高度，通过 android:layout_width 和 android:layout_height 属性设置。宽度和高度的属性有以下几种设置方式。

match_parent：表示将强制性扩展控件宽度至其父控件的宽度以显示全部内容。

wrap_content：表示将强制地扩展控件宽度以显示全部内容，控件的宽度会根据需要显示的内容进行调整，显示的内容多则控件宽，显示的内容少则控件窄。以 TextView 控件为例，设置为 wrap_content 将完整显示其内全部文本。

【注意】：fill_parent 和 match_parent 的含义相同，只不过 match_parent 更为贴切，从 Android2.2 开始两个词都可以使用，但 Google 推荐使用 match_parent，Android2.2 版本以下只支持使用 fill_parent。

2.2 关于 MainActivity.java 文件

每一个 Android 项目成功创建后，默认生成一个 Activity 文件 MainActivity.java，该文件位于项目的 src/java 文件夹的包文件夹中，主要用于实现界面的交互功能。

MainActivity.java 文件中默认生成的代码如表 2-2 所示。

表 2-2　MainActivity.java 文件中默认生成的初始代码

序号	代　码
01	package com.example.app0201;
02	import android.app.Activity;
03	import android.os.Bundle;
04	import android.view.Menu;
05	import android.view.MenuItem;
06	public class MainActivity extends Activity {
07	@Override
08	protected void onCreate(Bundle savedInstanceState) {
09	super.onCreate(savedInstanceState);
10	setContentView(R.layout.activity_main);
11	}
12	@Override
13	public boolean onCreateOptionsMenu(Menu menu) {
14	// Inflate the menu; this adds items to the action bar if it is present.
15	getMenuInflater().inflate(R.menu.menu_main, menu);
16	return true;
17	}
18	@Override
19	public boolean onOptionsItemSelected(MenuItem item) {
20	// Handle action bar item clicks here. The action bar will
21	// automatically handle clicks on the Home/Up button, so long
22	// as you specify a parent activity in AndroidManifest.xml.
23	int id = item.getItemId();
24	//noinspection SimplifiableIfStatement
25	if (id == R.id.action_settings) {
26	return true;
27	}
28	return super.onOptionsItemSelected(item);
29	}
30	}

由表 2-2 所示的代码可知，新建一个 Android Studio 项目时，系统为我们生成了一个 MainActivity.java 文件，同时自动导入了 2 个类（Activity、Bundle）和 2 个接口（Menu、MenuItem），MainActivity 类继承自 Activity 且重写了 onCreate()方法，该方法调用父类的 onCreate()方法，调用 setContentView()方法设置当前页面的布局文件为 activity_main，将布局

文件转换成 View 对象，显示在界面上。另外，第 13 行至第 17 行重定义 onCreateOptionsMenu 方法，第 19 行至第 29 行重定义 onOptionsItemSelected 方法。

① @Override

在重写父类的 onCreate 时，在方法前面加上@Override 标识，系统可以帮助检查方法的正确性。例如，public void onCreate(Bundle savedInstanceState){……}这种写法是正确的，如果写成 public void oncreate(Bundle savedInstanceState){……}，则编译器会输出如下提示信息：The method oncreate(Bundle) of type HelloWorld must override or implement a supertype method，以确保正确重写 onCreate 方法，因为 oncreate 应该为 onCreate。如果不加@Override 标识，则编译器将不会检测出错误，而是会认为新定义了一个方法 oncreate。

② android.app.Activity 类

android.app 为 Android 的包名称，该包中提供了高层的程序模型和基本的运行环境所需的类和接口。几乎所有的活动（activities）都是与用户交互的，Activity 类用于创建窗口，可以使用方法 setContentView(View)将 UI 放到窗口中。然而活动通常以全屏的方式展示给用户，也可以以浮动窗口或嵌入在另外一个活动中。以下两个方法是几乎所有的 Activity 子类都实现的。

onCreate(Bundle)：初始化活动（Activity），在这个方法中通常调用 setContentView()方法定义 UI，调用 findViewById()方法在 UI 中检索需要交互的小部件（widgets）。setContentView 指定由哪个文件指定布局（activity_main.xml），可以将这个界面显示出来，然后我们进行相关操作。

onPause()：处理当离开活动时要做的事情，用户做的所有改变应该在这里提交。

③ android.os.Bundle 类

android.os 为 Android 的包名称，该包提供了系统服务、消息传输、IPC 机制所需的类或包。Bundle 是主要用于在 Activity 之间传递数据的类，它保存的数据，是以 key-value(键值对)的形式存在的。该类提供了公有方法 public boolean containKey(String key)，实现了 Parceable 和 Cloneable 接口。

④ android.view.Menu 和 android.view.MenuItem 接口

android.view 为 Android 的包名称，该包提供了基础的用户界面接口框架所需的类或包。

Menu 是一个父接口，Android 用它来管理各种菜单项。它下面有两个子类，一个是 ContextMenu（表示一个上下文菜单），另一个是 SubMenu（表示一个子菜单）。它们都可以包含一个或多个的 MenuItem 菜单项。Menu 接口定义了两个方法来添加菜单项和添加子菜单，一个通过 add()方法添加菜单项，另一个通过 addSubMenu()方法来添加子菜单。注意我们一般不自己创建 menu，因为每个 Activity 默认都自带了一个 menu，我们要做的是为它加菜单项和响应菜单项的点击事件。

2.3 关于 AndroidManifest.xml 文件

文件 AndroidManifest.xml 是 Android Studio 项目的全局配置文件，是 Android 应用程序中最重要的文件之一，是每个 Android 程序中必须的文件，记录应用程序中所使用的各种控件。该文件提供了 Android 系统所需要的关于该应用程序的必要信息，即在该应用程序的任何代码运行之前系统所必须拥有的信息。另外当新添加一个 Activity 的时候，也需要在这个

文件中进行相应配置，只有配置好后，才能调用此 Activity。

AndroidManifest.xml 文件的结构、元素，以及元素的属性等方面的主要规则说明如下。

（1）元素：在所有的元素中只有<manifest>和<application>是必需的，且只能出现一次。如果一个元素包含有其他子元素，必须通过子元素的属性来设置其值。处于同一层次的元素，这些元素的说明是没有顺序的。

（2）属性：按照常理，所有的属性都是可选的，但是有些属性是必须设置的。那些真正可选的属性，即使不存在，其也有默认的数值项说明。除了根元素<manifest>的属性，所有其他元素属性的名字都是以 android:为前缀的。

（3）定义类名：所有的元素名都对应其在 SDK 中的类名，如果自定义类名，必须包含类的包名称，如果类与 application 处于同一数据包中，则可以直接简写为“.”。

（4）多数值项：如果某个元素有超过一个数值，这个元素必须通过重复的方式来说明其某个属性具有多个数值项，且不能将多个数值项一次性说明在一个属性中。

（5）资源项说明：当需要引用某个资源时，其采用如下格式：@[package:]type:name。例如<activity android:icon= "@drawable/icon " >。

（6）字符串值：如果字符中包含有字符“\”，则必须使用转义字符“\\”。

AndroidManifest.xml 文件中的通常设置如表 2-3 所示。

表 2-3　AndroidManifest.xml 文件中的通常设置

属 性 名 称	含　义	取　值
android:launchMode	启动模式	standard、singleTop、singleTask、singleInstance
android:screenOrientation	屏幕	landscape（横屏）、portrait（竖屏）
android:label	标题名称	直接写字符，或引用 XML 文件中的@string
android:name	Activity 类名	通常 package 加 name 等于 Activity 类带包名的全称

AndroidManifest.xml 文件默认生成的代码如表 2-4 所示。

表 2-4　AndroidManifest.xml 文件默认生成的代码

序号	代　码
01	<?xml version="1.0" encoding="utf-8"?>
02	<manifest xmlns:android="http://schemas.android.com/apk/res/android"
03	package="com.example.app0201" >
04	<application
05	android:allowBackup="true"
06	android:icon="@mipmap/ic_launcher"
07	android:label="@string/app_name"
08	android:theme="@style/AppTheme" >
09	<activity
10	android:name=".MainActivity"
11	android:label="@string/app_name" >
12	<intent-filter>
13	<action android:name="android.intent.action.MAIN" />

续表

序号	代　码
14	<category android:name="android.intent.category.LAUNCHER" />
15	</intent-filter>
16	</activity>
17	</application>
18	</manifest>

AndroidManifest.xml 文件各层结点解析如下：

（1）第一层（Manifest）的主要属性

```
<manifest  xmlns:android="http://schemas.android.com/apk/res/android"
        package=" com.example.app0201"
        ……
</manifest>
```

xmlns:android 定义 android 命名空间，一般为 http://schemas.android.com/apk/res/android，这样使得 Android 中各种标准属性能在文件中使用，提供了大部分元素中的数据。

package 指定本应用程序内 Java 主程序包的包名，它也是一个应用进程的默认名称。

（2）第二层（application）的主要属性

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    ……
</application>
```

一个 AndroidManifest.xml 中必须含有一个 Application 标签，这个标签声明了每一个应用程序的控件及其属性，例如 icon、label、permission 等。

android:allowBackup 设置应用程序是否可以通过 adb 命令进行备份整个应用的数据。

android:icon 声明应用程序的图标，图片一般都放在 drawable 文件夹下，Android 智能终端中看到的图标就是该属性指定的。

android:label 给当前的 ViewGroup 设置一个标签。

android:theme 定义了一个默认的主题风格给所有的 activity，当然也可以在自己的 theme 里面去设置它，与网页中的 style 有点类似。

（3）第三层（activity）的主要属性

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name" >
</activity>
```

<activity>标签位于<application>标签下方，一个应用程序可以有多个 Activity，因此可能会出现多个<activity>标签，该标签也包含很多信息。

android:name 用于指定 Activity 类的名称，例如 com.example.app0201.MainActivity。

android:label 用于指定应用程序在桌面上的名称和主 activity 的 title。

activity 和 application 里都可以设置 android:label 标签，activity 的优先级高于 application，

也就是说两者都设置这个标签的话，activity 设置的 label 值覆盖 application 设置的值。

当 application 里设置了此标签，其他 activity 没有设置的情况下，应用程序名在桌面上的名字和所有 activity 的 title 是这个设置的标签。

当 application 里设置了此标签，主 activity 中也设置了此标签，则应用程序名和主 activity 的 title 都是主 activity 中设置的标签，其他非主 activity 的 title 如果没有自己设置此标签，还是使用 application 中设置的标签，如果其他非主 activity 也设置了此标签，则其 title 就是自己设置的这个标签。

（4）第四层（intent-filter）的主要属性

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

常见的 android:name 值为 android.intent.action.MAIN，表明该 activity 是作为应用程序的入口，当应用程序存在多个 Activity 时，该属性决定了哪个 Activity 最先被启动。

常见的 android:name 值为 android.intent.category.LAUNCHER，表示应用程序在上层的启动程序列表里显示。

2.4 Android 应用程序的样式和主题设置

Android 应用程序的样式设置包括样式定义、单个 view 样式设置、 全局样式设置、样式继承关系等。

1．样式定义

Android 的样式定义在 res/values/styles.xml 文件中，类似 Web 网页中将样式定义在某个 CSS 文件中，但 Android 的 styles.xml 是自动加载的，不需要手动 import 或 link。

如下所示的代码是一组样式的定义。

```
<resources>
    <style name="DefaultFontStyle">     <!-- 全局字体样式-->
        <item name="android:textSize">18px</item>
        <item name="android:textColor">#0000CC</item>
    </style>
    <!-- 全局背景色-->
    <style name="DefaultBgColor" parent="@style/DefaultFontStyle">
        <item name="android:background">#F2F2F2</item>
    </style>
    <!-- 全局样式-->
    <style name="DefaultStyle" parent="@style/DefaultBgColor"> </style>
</resources>
```

Android 的样式定义是通过 style 标签完成的，通过添加 item 元素设置不同的属性值。样式可以通过设置 parent 进行继承。上面的 DefaultBgColor 继承自 DefaultFontStyle，而 DefaultStyle 又继承自 DefaultBgColor，这样 DefaultStyle 就有了字体大小颜色、背景色的属性了。

2．设置单个控件样式

对于 TextView，样式设置的代码如下。

```
<TextView android:text="OK"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18px"
    android:textColor="#0000CC" />
```

也可以引用前面定义的样式，代码如下。

```
<TextView android:text="OK"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    style="@style/DefaultStyle" />
```

设置控件的 style 属性进行样式调用，推荐使用此种方式将样式和布局分离。

3．全局样式设置

在 Web 前端编程中，可以使用 CSS 样式文件设置全局的样式，也可以设置单个标签的样式。Android 中我们同样可以办到，只是这种全局样式被称作主题 theme，例如对于整个应用默认字体都要 18px，颜色为#0000CC，背景色为#F2F2F2，我们可以通过在 AndroidManifest.xml 设置 application 的 android:theme 属性完成，代码如下：

```
android:theme="@style/AppTheme"
```

引用主题样式使用 android:theme，主题的设置也可以在代码中通过 setTheme(R.id.xx) 完成。

4．样式继承关系

Android 的样式采取和 CSS 中一样的覆盖、继承原则，和面向对象的子类覆盖父类属性、子类继承父类属性值的原则是一样的。

如果一个 TextView 自己设置了样式，它的 ViewGroup 设置了样式，activity 设置了主题，application 设置了主题。它会先读取自己样式的值，对于自己没有的样式向上查找第一个找到的值即为要采取的值。依次读取的顺序为 View 自己的样式→上一层 ViewGroup 的属性值→上上层 ViewGroup 的属性值→…→activity 主题→application 主题。

2.5 关于 Android 系统的包

Android 提供了扩展的 Java 类库，类库分为若干个包，每个包中包含若干个类。

新建一个 Android Studio 项目时，会默认生成一个 Activity 文件 MainActivity.java，该文件中会自动导入 2 个类（Activity、Bundle）和 2 个接口（Menu、MenuItem），代码如下所示。

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

在 Android 中，各种包写成 android.*的方式，重要包的描述如下所示：

android.app：提供高层的程序模型、提供基本的运行环境。

android.os：提供系统服务、消息传输和 IPC 机制。

android.view：提供基础的用户界面接口框架。

android.content：包含各种对设备上的数据进行访问和发布的类。

android.database：通过内容提供者浏览和操作数据库。

android.graphics：提供底层的图形库，包含画布、颜色过滤、点、矩形，可以将它们直接绘制到屏幕上。

android.location：提供定位和相关服务的类。

android.media：提供一些类，管理多种音频、视频的媒体接口。

android.net：提供帮助网络访问的类。

android.opengl：提供 OpenGL 的工具。

android.provider：提供类访问 Android 的内容提供者。

android.telephony：提供与拨打电话相关的 API 交互。

android.util：提供涉及工具性的方法，例如时间日期的操作。

android.webkit：提供默认浏览器操作接口。

android.widget：包含各种 UI 元素（大部分是可见的）在应用程序的屏幕中使用的类。

本书各单元的各项任务中各个类及相关包的引用详见附录 C。

2.6 相关问题剖析

1．如何区分 Android Studio 中的 Project 和 Module

在 Android Studio 中，Project 的真实含义是工作空间，Module 为一个具体的项目。Modules 是“一种独立的功能单元，可以运行、测试并且独立调试”，和 Eclipse project 的概念有点像。android studio 中的项目是 eclispe 中 workspace 的意思，eclispe 中的项目在 android studio 中是一个 module。

每一个 Module 需要有属于自己的 Gradle build file（当新建一个 Module 时会自动生成，当导入一个 Eclipse 项目时需自己创建）。这些 Gradle 文件包含了一些很重要的内容。

在 Eclipse 中，可以同时对多个 Eclipse 的 Project 进行同时编辑，这些 Project 在同一个 workspace 之中。在 Android Studio 中，我们则同时对多个 Android Studio 的 Module 进行编辑，这些 Module 在同一个 Project 之中。Eclipse 的 Project 等同于 Android Studio 的 Module，Eclipse 的 workspace 等同于 Android Studio 的 Project。

Android Studio 创建项目的过程，其实就是 Eclipse 创建项目过程的细分化。Eclipse 许多在一个页面设置的内容，Android Studio 拆分成了多个页面，因此，创建项目的过程其实并不复杂。

2．如何在 Android Studio 开发环境中打开 Android 项目文件所在的文件夹？

在 Android Studio 的 Project 面板中，右击项目文件，在弹出的快捷菜单中选择“Show in Explorer”命令，如图 2-3 所示，就可以打开项目文件所在的文件夹了。

3．Android 程序如何获取界面上的控件并在窗口中显示

（1）在 activity_main.xml 配置文件中，为界面上的控件增加 android:id 属性，代码如下所示：

```
<TextView
   android:id="@+id/device_id"
   ……    />
```

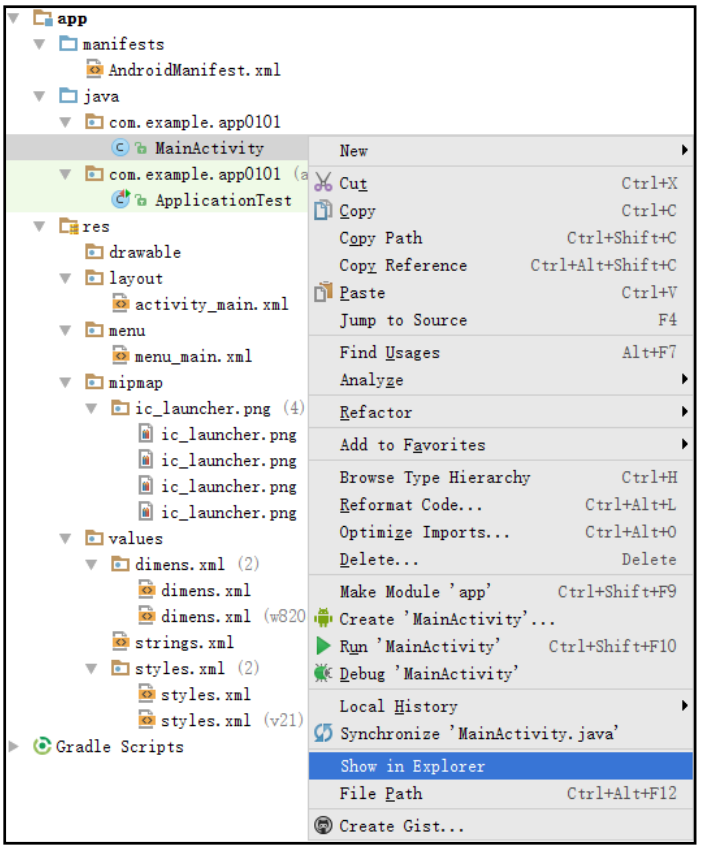


图 2-3　在快捷菜单中选择“Show in Explorer”命令

（2）在 Activity 类中通过 findViewById()方法获取对应的 android:id

获取 activity_main.xml 中配置的控件的代码如下：

```
TextView tv = (TextView)findViewById(R.id.device_id);
```

如果想让布局文件显示在当前窗口中，需要在 MainActivity 中的 onCreate()方法中通过代码“setContentView(R.layout.布局文件名称);”将布局文件加载到 View 对象中。这样当程序运行时，才能在界面看到编写好的布局。

【注意】：这里我们给这个 TextView 赋了一个 id 值：android:id="@+id/device_id"。这里的“@+id”表示创建一个新的 id 值，这时候 Android 平台开发的 aapt 工具（Android Asset Packaging Tool）就会帮我们生成一个字段值，让我们可以通过 R.id. device_id 来查找出这个对象。如果是“@id”则表示这个是对 ID 值的引用，aapt 工具不会帮我们生成一个字段值来获取这个 id。

4．在 Android 的布局文件 activity_main.xml 中“@+id/username”与"@id/username" 两者有何区别

分析如下所示的布局文件 activity_main.xml 的代码

```
<TextView
        android:id="@+id/xyz"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
```

其中包含了代码 android:id="@+id/xyz"和 android:text="@string/hello_world"。

Android 中的控件需要用一个 int 类型的值来表示，这个值也就是控件标签中的 id 属性值。id 属性只能接受资源类型的值，也就是必须以@开头的值，例如，@id/abc、@+id/xyz 等。如果在@后面使用“+”，表示当修改完某个布局文件并保存后，系统会自动在 R.java 文件中生成相应的 int 类型变量，变量名就是“/”后面的值，例如，@+id/xyz 会在 R.java 文件中生成 int xyz = value，其中 value 是一个十六进制的数。如果 xyz 在 R.java 中已经存在同名的变量，就不再生成新的变量，而该控件会使用这个已存在的变量的值。也就是说，如果使用@+id/name 形式，当 R.java 中存在名为 name 变量时，则该控件会使用该变量的值作为标识。如果不存在该变量，则添加一个新的变量，并为该变量赋相应的值（不会重复）。android:id="@id/xyz"是表示 TextView 的 id 是 xyz，然后这个就可以通过 findViewById（R.id.xyz）获得一个 TextView 对象。

android:text="@string/hello_world"是说明在 res 目录下建立了一个名为 string 的 xml 文件，其中包含一个名为 hello_world（其值可以是任意的一段话）的 string 数组。这句话的意思就是将本 textView 的内容设置为 hello_world 所代表的值。



【任务 2-1】熟悉 Android Studio 的组成结构

【任务描述】

创建 Android Studio 项目 App0201，熟悉 Android Studio 的组成结构及功能。

【知识索引】

（1）Android Studio 主界面的基本组成。

（2）Project 面板的各级目录结构以及主要文件。

（3）Android Studio 的实时布局、富布局编辑器。

（4）Android StudioLint 工具。

【实施过程】

1．创建 Android 项目 App0201

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0201。

2．熟悉 Android Studio 主界面的基本组成

Android Studio 主界面的基本组成如图 2-4 所示。

Android Studio 主界面显示了使用 Android Studio 时经常接触到的功能面板。

（1）Project 面板。Project 面板用于浏览项目文件，该面板中会显示当前的所有 module。

（2）Build Variants 面板。Build Variants 面板用于设置当前项目的 Build Variants，所有的 Module 默认都会有 release 和 debug 两种选项。

（3）Android DDMS 面板。Android DDMS 面板的功能类似于 Eclipse 中的 Logcat，但是比其多了一些常用功能，例如截图，查看系统信息等。

（4）编辑区。编辑区用于编辑文件。

（5）Gradle tasks 面板。Gradle tasks 面板用于显示 Gradle 任务列表，例如 assemble、assembleRelease、assembleDebug、build、check、clean、lint 等常用任务，双击即可执行 Gradle 任务。

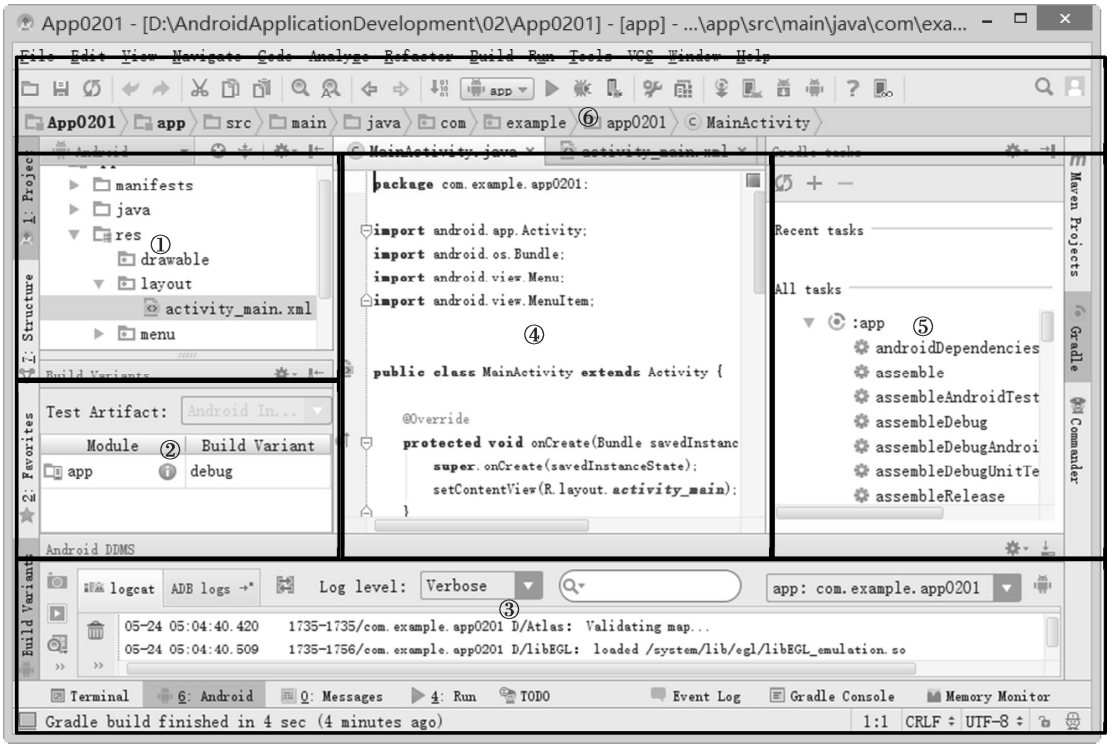


图 2-4　Android Studio 主界面的基本组成

3．熟悉 Project 面板的各级目录结构以及主要文件

Android Studio 主界面默认的 Project 面板显示的视图类型为 Android 类型，如图 2-5 所示。通过单击左上角的标签可以进行切换，在视图类型下拉列表选择“Project”选项，如图 2-6 所示，即可切换为 Project 类型，如图 2-7 所示。

（1）src 文件夹与 MainActivity.java 文件

src（source code）文件夹是存放项目源代码的文件夹，每一个 Android 项目成功创建后，默认生成的 Activity 文件 MainActivity.java 就位于 src/java 文件夹的包文件夹中。包文件夹（例如 com/example/app0201）也是位于 src 文件夹中。

在 src 下是主程序类。如果在建立项目时，选择并填写了 Create Activity 时，会自动生成名为填写内容的，继承自 android.app.Activity 的类，在类中重写了 onCreate()方法。方法中的 setContentView 为设置这个 Activity 的显示布局(R.layout.main)，布局文件在 res/layout 下。R.layout.main 实际上是指 res/layout/main.xml 布局文件。

（2）res 文件夹

res 文件夹为存放资源的文件夹，它包含项目中资源文件并将编译进应用程序。向该文件夹中添加资源时，会被 R.java 文件自动记录。新建一个项目，res 文件夹下会自动添加 5 个子文件夹：drawabel、layout、menu、mipmap 和 values。

drawabel 子文件夹中存放待用的图片文件(*.png、*.jpg)，layout 子文件夹中存放界面布局文件 activity_main.xml，menu 子文件夹存放菜单资源文件，通过编辑这些菜单文件可以直接控制菜单项的显示，mipmap 子文件夹存放应用程序待用图标文件。

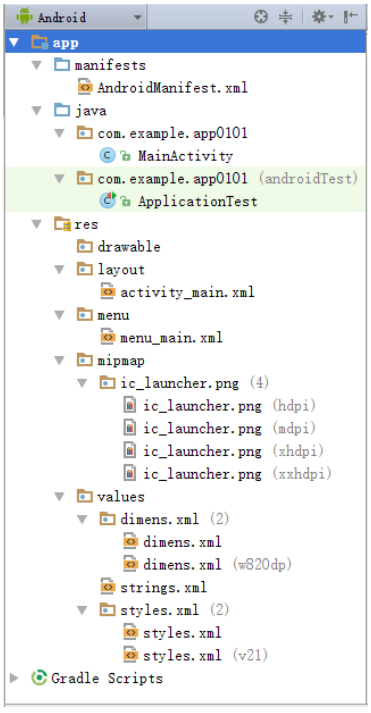


图 2-5　Android Studio 的 Project 面板的“Android”视图

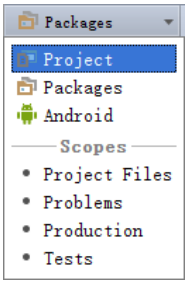


图 2-6　改变 Project 面板的视图类型

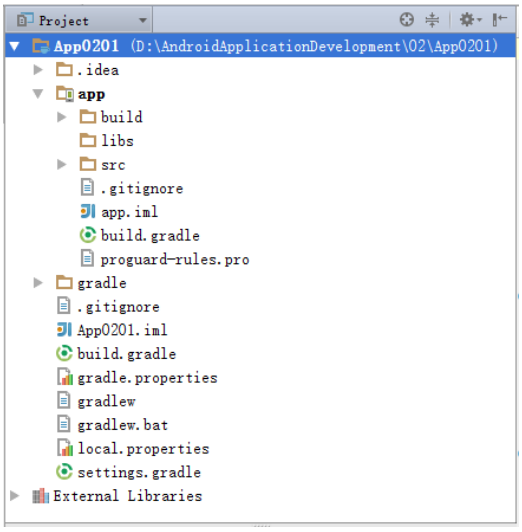


图 2-7　Android Studio 的 Project 面板的“Project”视图

values 子文件夹可以存放多个*.xml 文件，一般包括 dimens.xml、strings.xml、styles.xml 等配置文件，还可以存放不同类型的数据。dimens.xml 文件用于定义尺寸常量值，示例代码如下所示。

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

strings.xml 用于定义字符串常量值，示例代码如下所示。

```
<resources>
    <string name="app_name">App0201</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

styles.xml 用于定义应用程序的 style，示例代码如下所示。

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="android:Theme.Holo.Light.DarkActionBar">
        <!-- Customize your theme here. -->
    </style>
</resources>
```

4．熟悉 Android Studio 的实时布局（Live Layout）

Android Studio 中的实时布局功能允许在无需将应用程序运行在设备或者模拟器中的前提下，直接预览应用的用户界面。实时布局是一款极为强大的工具，能够帮助开发者节约大量时间。在实时布局的帮助下，查看应用程序用户界面的任务变得轻松而且快捷。



图 2-8　Android Studio 的实时布局

要使用实时布局，需要双击对应 XML 布局文件并选择编辑工作区下方的“Text”标签。接下来选择工作区右侧的“Preview”标签来预览当前布局，如图 2-8 所示。对 XML 布局作出的任何变更都会直接反映在右侧的预览窗口当中。

实时布局功能带来了诸多优势，例如可以在 Preview 面板的第 1 项菜单中选择选项，从而对当前正在使用的 XML 布局随意作出调整。Preview 面板中的第 2 项菜单允许变更显示在 Preview 面板内的设备大小。第 3 项菜单的作用则是调整设备在 Preview 面板中的朝向，这样我们就能更轻松地查看同一套布局方案在不同显示模式（纵向与横向）下的效果与主题。Preview 面板中的第 4 项菜单允许我们轻松访问 Activity 或者布局所使用的个别片段。Preview 面板还允许大家变更在实时布局中使用的语言，从而轻松预览不同语言在布局方案中的显示效果。最右边的菜单的作用则是调整 API 版本。Preview 面板中还包含多项控制机制，例如对布局进行缩放、重新 Preview 面板或者截取当前屏幕等。

5．熟悉 Android Studio 的富布局编辑器

Android Studio 提供一套富布局编辑器，可以在其中随意拖曳各类用户界面控件，还可以在多屏幕配置中同时查看多种布局的显示效果。

在编辑区的底部可以看到两个标签，分别是“Design”与“Text”。单击“Text”标签后编辑器将被激活，就能对当前选定的布局方案作出变更。单击“Design”标签则会激活另一套编

辑器内容，其中显示出布局的预览效果。要向布局当中添加其他功能性控件，我们只需将其从布局左侧的控件列表中拖出并放入布局内即可。

6．熟悉 Android Studio 的模板

Android Studio 为开发人员提供多种模板选项，从而大大提升开发速度。这些模板能自动创建 Activity 以及必要的 XML 文件。还可以利用这些模板创建出较为基础的 Android 应用程序，并将其运行在实体设备或者模拟器当中。

在 Android Studio 中，可以在创建新的 Activity 时一同创建出对应模板。右击窗口左侧项目浏览器中的“package name”，并在菜单中选择命令“New”，而后从选项列表中选择“Activity”，即可显示 Android Studio 的模板清单，如图 2-9 所示，其中包括 Blank Activity、Fullscreen Activity、Tabbed Activity 等多种模型。

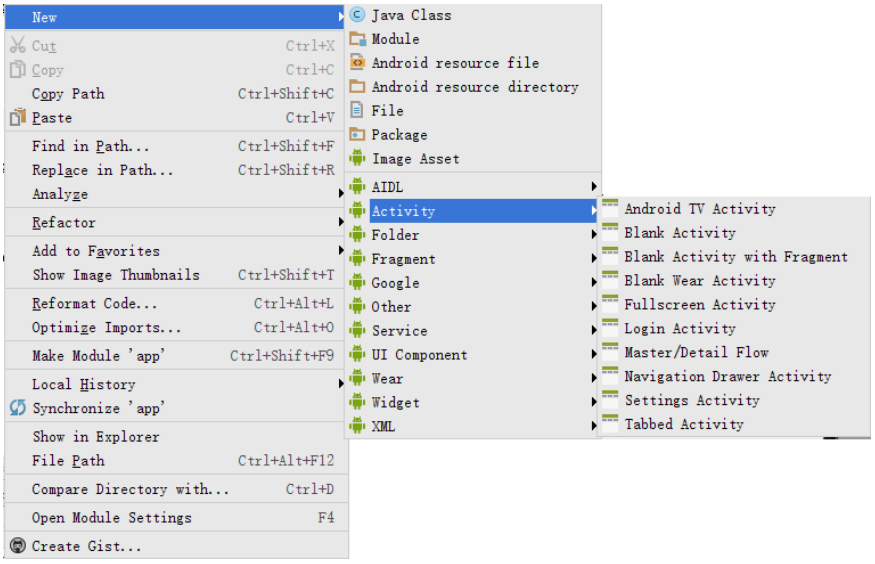


图 2-9　Android Studio 的模板列表

7．熟悉 Android StudioLint 工具

Android Studio 中提供的 Android Lint 是一款静态分析工具，它负责对项目源代码加以分析，它能够检测出应用程序中的潜在漏洞以及其他可能被编译器所忽略的其他代码问题。

Android Lint 的优势在于：它能帮助我们重视警告或报错信息的出现原因，从而更轻松地修复或者解决这些问题。我们要养成重视使用 Android Studio Lint 工具的好习惯，这能帮助我们准确检测到项目当中存在的潜在问题，Lint 工具甚至能告诉我们应用程序中是否存在重复的图片或者编译内容。

当 Android Studio 完成对项目的检测之后，它会在窗口底部显示出分析结果。除了 Android Lint 之外，Android Studio 还提供一系列其他检查功能，只需双击某个已经发现的问题，系统就会帮助大家定位到对应文件中存在问题的位置。

Android Studio 提供了强大的代码分析功能，使用这个功能可以发现我们项目中的问题。然后在“Analyze”菜单中选择“Inspect Code”命令，如图 2-10 所示。

弹出如图 2-11 所示的“Specify Inspection Scope”对话框，单击【OK】按钮即可开始分析代码，分析完毕即会显示分析结果。

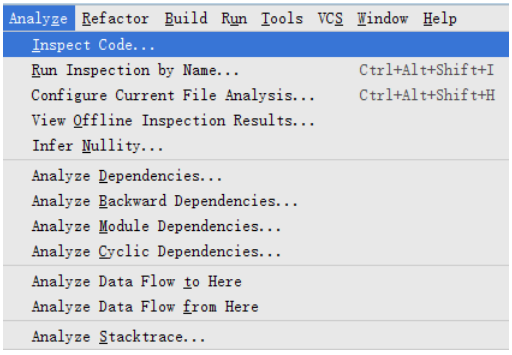


图 2-10　在“Analyze”菜单中选择“Inspect Code”命令

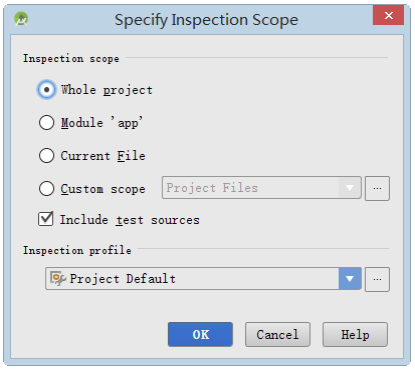


图 2-11　“Specify Inspection Scope”对话框

【任务 2-2】Android Studio 项目中的模块操作

【任务描述】

在 Android 项目 App0201 中完成以下操作：

（1）在 Android Studio 项目 App0201 中添加一个 Module。

（2）将模块 Module 重命名。

（3）删除模块 Module。

【知识索引】

（1）在 Android Studio 项目中添加 Module。

（2）模块的重命名与删除。

【实施过程】

1．在项目 App0201 中创建一个模块 App0202

打开已创建的项目 App0201，然后在 Android Studio 主界面的菜单“File”中选择命令“New Module”，打开“Create New Module”对话框，“New Module”界面如图 2-12 所示。

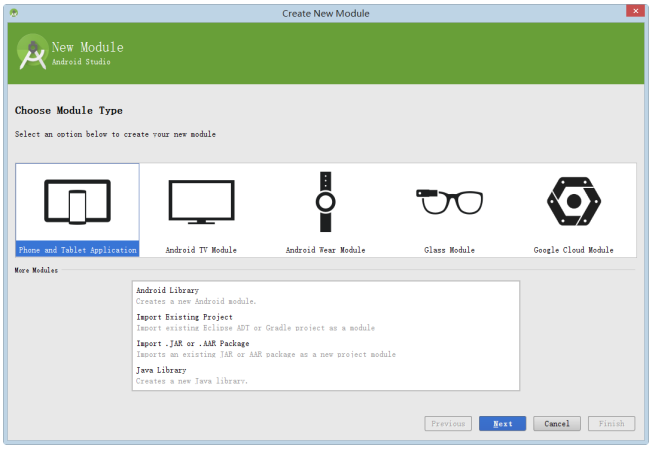


图 2-12 【Create New Module】之“New Module”界面

“Module Type”有多个选项，选择左边第 1 项“Phone and Tablet Application”，对话框下

方也有多个选项，其中“Android Library”表示“创建一个 Android Library 的 Module”；“Import Existing Project”表示“导入其他项目为一个 Module”，导入的项目可以是 Eclipse ADT 项目，也可以是 Gradle 项目（Gradle 只是项目构建工具，Eclipse 中也是可以使用的）；“Import .JAR or .AAR Package”表示“导入 JAR 或者 AAR 文件为一个 Module”；“Java Library”表示“创建一个 Java 的 Module”，主要用于编写 Java 的工具包。

然后单击【Next】按钮进入下一步，设置模块名称为“app0202”，如图 2-13 所示。

后面的步骤与创建一个 Android 项目的步骤类似，在此不再赘述。新添加一个模块 app0202 的【Project】面板如图 2-14 所示。

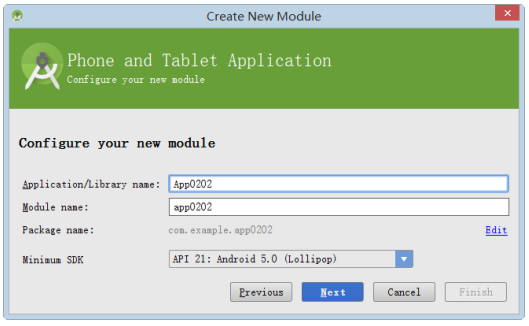


图 2-13　设置模块名称

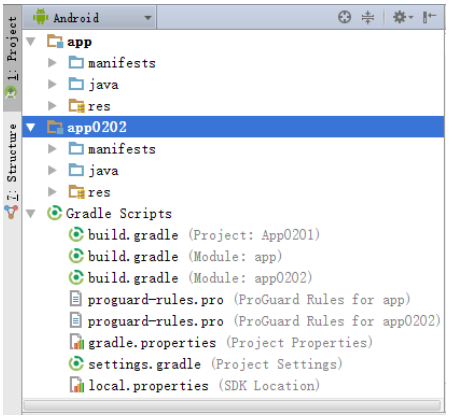


图 2-14　新添加一个模块 app0202 的【Project】面板

2．模块 Module 重命名

在【Project】面板中右击需重命名的模块名称，在弹出的快捷菜单中依次选择“Refactor”→“Rename”命令，如图 2-15 所示。打开“Rename Module”对话框，在该对话框的文本框中输入新的名称 app02，如图 2-16 所示，然后单击【OK】按钮即可。

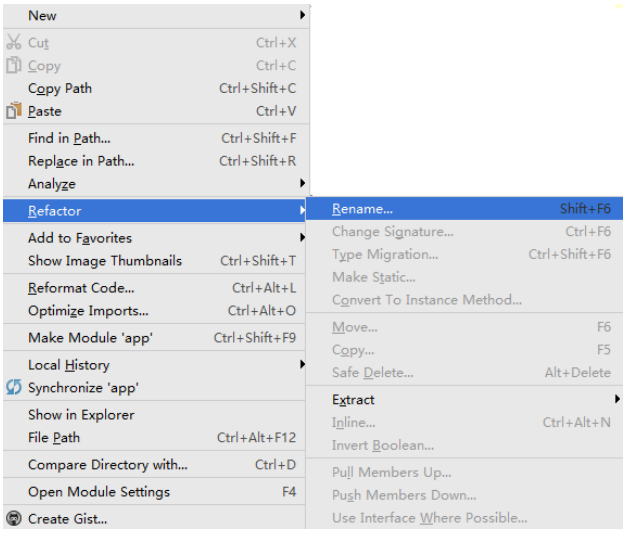


图 2-15　在快捷菜单中选择“Refactor”→“Rename”命令

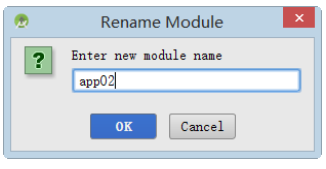


图 2-16　“Rename Module”对话框

3．删除模块 Module

当想在 Android Studio 中删除某个 module 时，大家习惯性的做法都是选中要删除的 module，右击，在弹出的快捷菜单中寻找“Delete”命令。但是在 Android Studio 中选中 module，右击，在弹出的快捷菜单会发现没有“Delete”命令，为什么会没有“Delete”命令？其主要原因是 Android Studio 对 module 做了一个保护机制，就是一个 module 是不能随意删除的，要删除必须先将 module 从 module 列表中移除。

在 Android Studio 主界面的菜单“File”中选择命令“Project Structure”，打开“Project Structure”对话框，在该对话框中选择模块“app0202”，然后单击上面的减号按钮，如图 2-17 所示。这时候会弹出“Remove Module”的对话框，如图 2-18 所示，在该对话框中单击【Yes】按钮，然后再单击“Project Structure”对话框中的【OK】按钮即可。

执行完这步操作后，再次选中 app0202，右击，在弹出的快捷菜单中就会发现“Delete”命令出现了，如图 2-19 所示。在快捷菜单中选择“Delete”命令就可以把 module 删除掉了。

【注意】：这里选择“Delete”命令后删除模块后，硬盘上也同样被删除了。

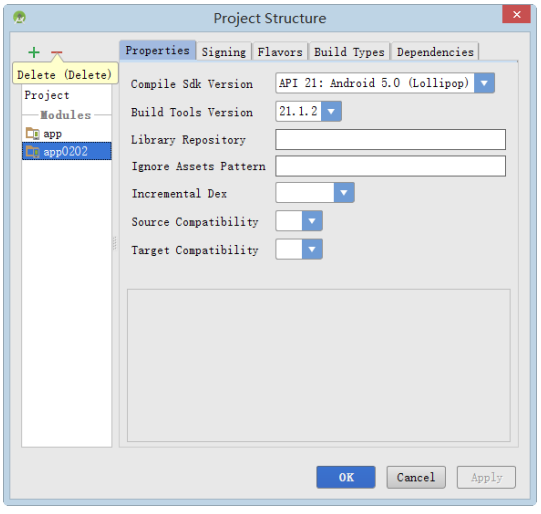


图 2-17 “Project Structure”对话框

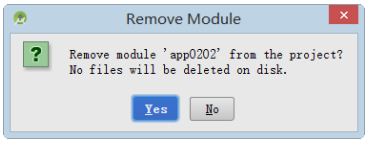


图 2-18 “Remove Module”对话框

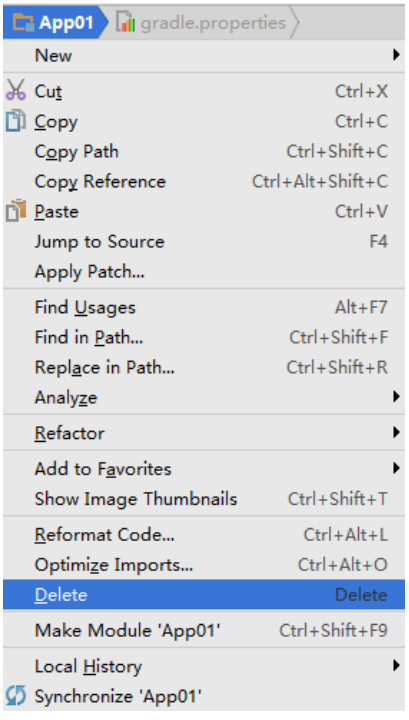


图 2-19 在快捷菜单中选择“Delete”命令

【任务 2-3】 Android Studio 开发环境的个性化设置

【任务描述】

在 Android Studio 开发环境中完成以下各项设置：

（1）主题风格、代码和系统界面字体。

（2）编码格式、快捷键。

（3）显示代码行数、自动导入包、自动编译功能。

（4）国际化功能。

【知识索引】

（1）主题风格设置。

（2）代码和系统界面字体设置。

（3）编码格式和快捷键的设置。

（4）显示代码行数、自动导入包和自动编译功能的设置。

（5）国际化功能的设置。

【实施过程】

1．设置主题风格

在 Android Studio 主界面中选择菜单命令“File”→“Settings”，打开“Settings”对话框，在该对话框左侧设置选项列表框中选择“Appearance”选项，右侧“Theme”下拉列表中有 3 个选项，分别为 Darcula、IntelliJ 和 Windows，如图 2-20 所示，根据个人偏好选择一个主题，然后单击【Apply】按钮或者【OK】按钮，改变 Android Studio 开发环境的主题。

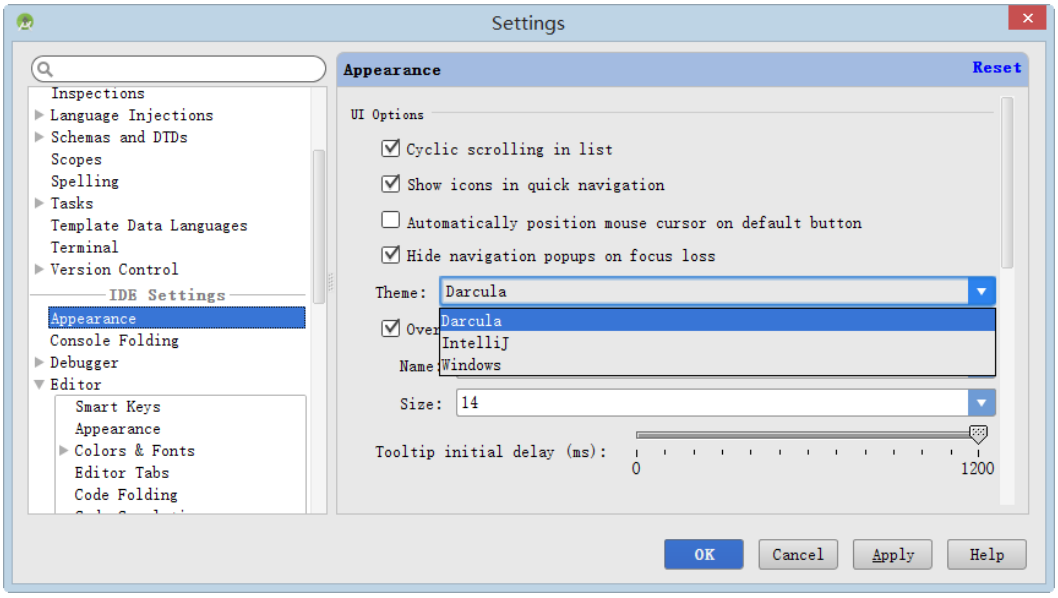


图 2-20　在“Settings”对话框中设置主题

2．设置代码的字体

在“Settings”对话框左侧搜索框中输入“Font”，然后找到“Colors & Fonts”下的“Font”选项，可以看到默认字体大小是 12，但是无法修改，如图 2-21 所示，此时“Scheme name”为“Default”。单击【Save As】按钮，弹出“Sava Color Scheme As”对话框，在该对话框的“Name”文本框中输入新的方案名称，这里输入“scheme01”，如图 2-22 所示，单击【OK】按钮即可。

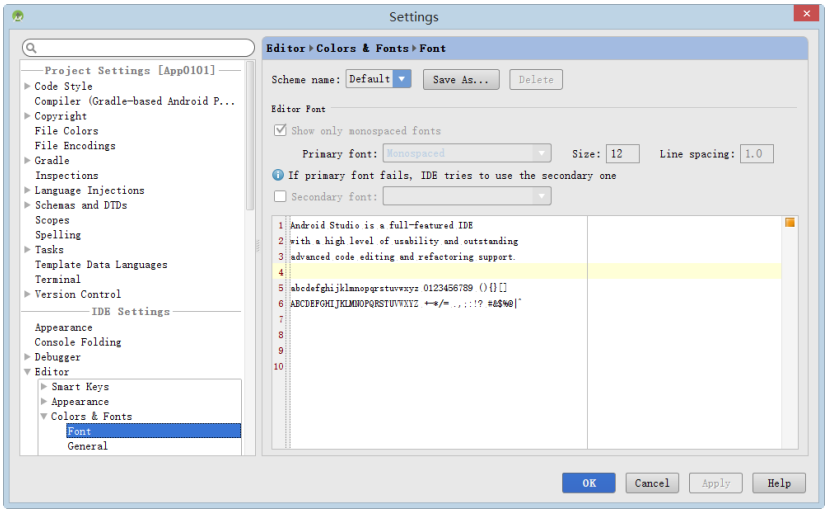


图 2-21 【Settings】对话框的“Font”默认值

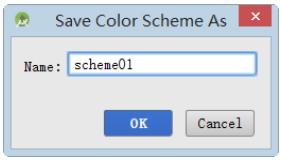


图 2-22 【Sava Color Scheme As】对话框

然后“Scheme name”自动设置为“scheme01”，并且可以修改字体大小和字体样式了，这里设置“Size”为 14，如图 2-23 所示，单击【OK】按钮，返回 Android Studio 主界面后，发现编辑区代码的字体大小改变了。

3．设置系统界面的字体

在“Settings”对话框左侧设置选项列表框中选择“Appearance”选项，右侧选中复选框“Override default fonts by(not recommended)”，该复选框下方的字体选择“宋体”，字体大小选择“14”，如图 2-24 所示，单击【OK】按钮，返回 Android Studio 主界面后，发现非编辑区界面（例如左侧 Project 面板目录结构）的字体大小改变了。

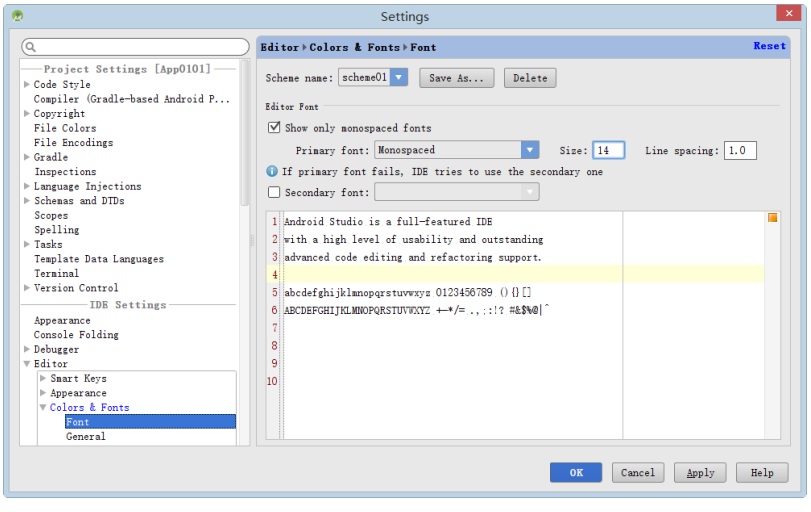


图 2-23 在“Settings”对话框改变代码的字体

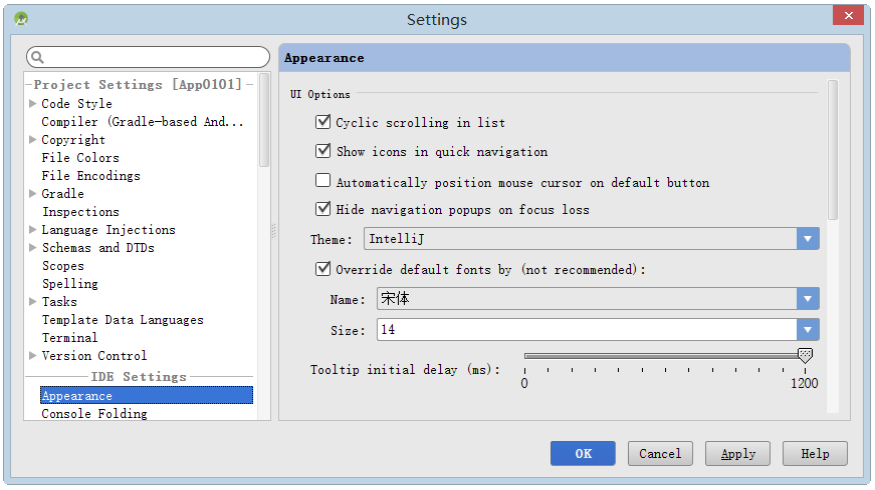


图 2-24　设置非编辑区界面的字体

4．设置编码格式

在“Settings”对话框左侧设置选项列表框中选择“File Encodings”选项，右侧可以分别设置 IDE Encoding、Project Encoding 和 Default encoding。在 Encoding 下拉列表中选择一种合适的编码格式，推荐都设置为 UTF-8，如图 2-25 所示，然后单击【OK】按钮即可。

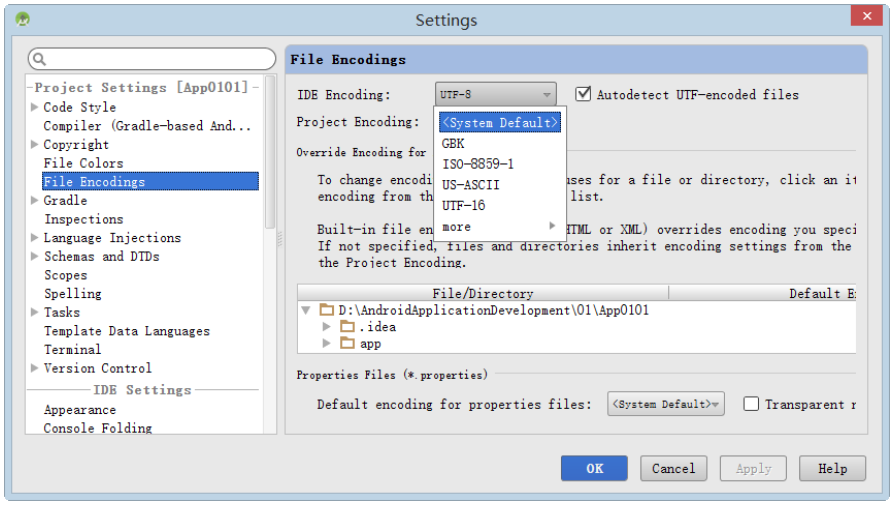


图 2-25　在“Settings”对话框设置编码格式

5．设置快捷键

Android Studio 的快捷键很灵活，不仅有自己的一套快捷键系统，而且为了方便 Eclipse 的使用者，还可以沿用 Eclipse 的快捷键习惯，这样我们都能很快上手。在“Settings”对话框左侧设置选项列表框中选择“Keymap”选项，在右侧的 Keymaps 下拉列表中选择一种适用自己的快捷键方式，如图 2-26 所示，然后单击【OK】按钮即可。

6．设置显示代码行数

在 Android Studio 代码编辑区域的左侧右击，在弹出的快捷菜单中选中“Show Line Numbers”选项即可，如图 2-27 所示。

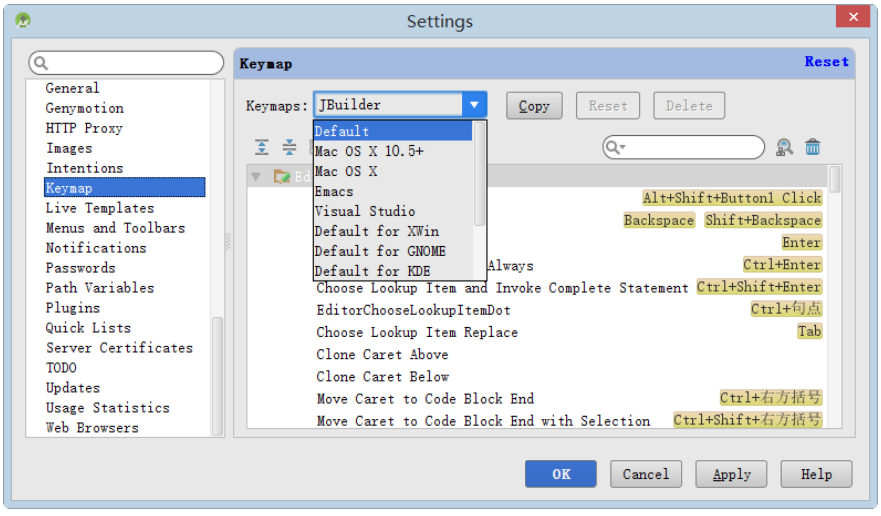


图 2-26　在“Settings”对话框设置快捷键

7．设置自动导入包

每次引用一些类的时候必须要导入包，而 Android Studio 可以通过设置自动导入包。在“Settings”对话框左侧设置选项列表框中选择“Auto Import”选项，在右侧将“Java”区域的复选框全部选中，如图 2-28 所示。然后单击【OK】按钮即可。

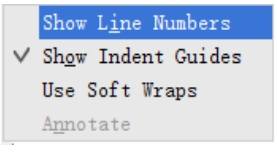


图 2-27　设置显示代码行数的快捷菜单

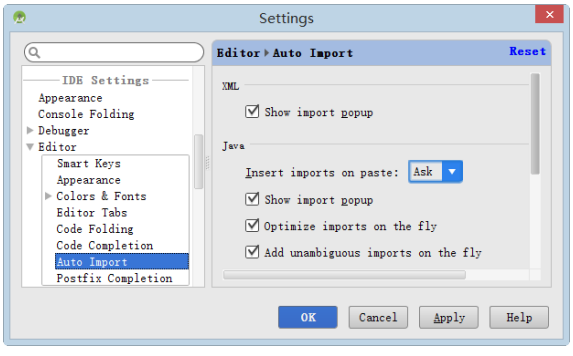


图 2-28　在“Settings”对话框设置自动导入包

8．设置自动编译功能

Android Studio 具有自动编译功能，自动编译意味着每次运行都能缩短很多时间。在“Settings”对话框左侧设置选项列表框中选择“Compiler”选项，右侧选中“Make project automatically”复选框，如图 2-29 所示。然后单击【OK】按钮即可。

9．设置国际化功能

在 Android Studio 主界面的【Project】面板中右击“res”文件夹，在弹出的快捷菜单中依次选择命令“New”→“Android resource directory”，如图 2-30 所示。

在弹出的“New Resource Directory” 对话框中，“Resource type”下拉列表框中选择“values”，“Available qualifiers”下拉列表框中选择“Language”，如图 2-31 所示。

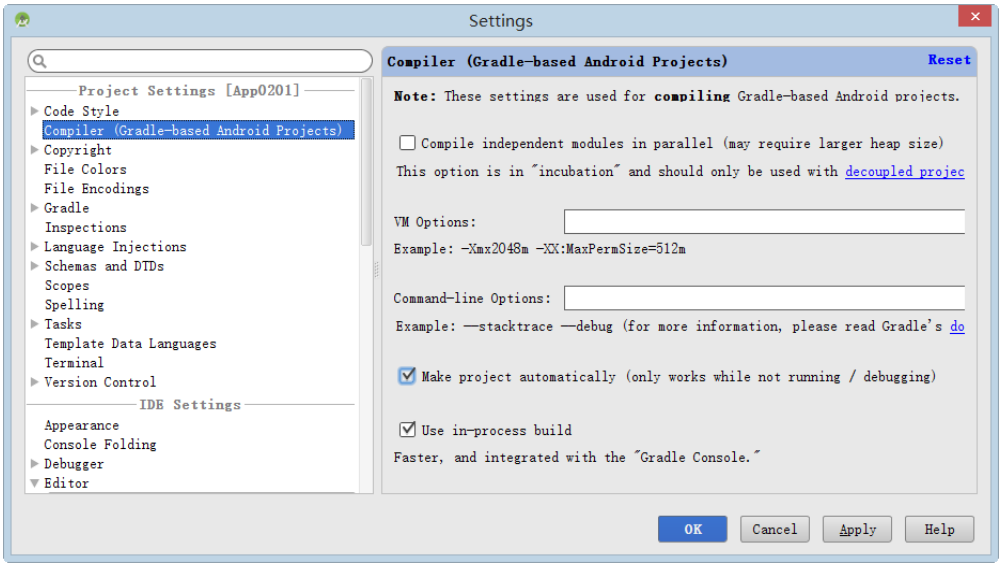


图 2-29　在“Settings”对话框设置自动编译功能

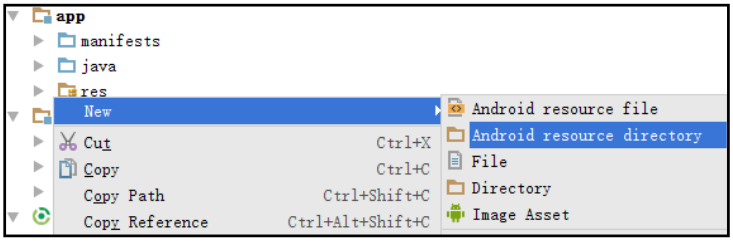


图 2-30　在快捷菜单中选择“Android resource directory”命令

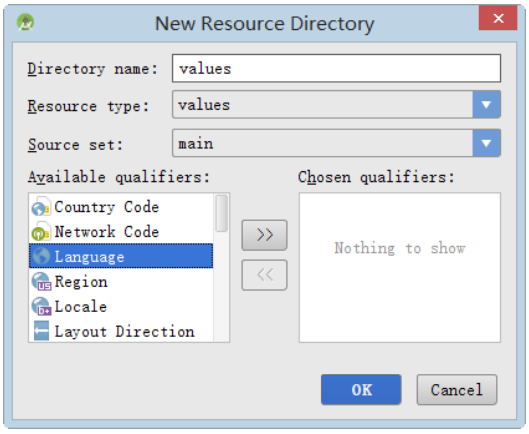


图 2-31　在【New Resource Directory】中选择“values”和“Language”

单击 » 按钮，然后在“Language”下拉列表中选择“zh:Chinese”，如图 2-32 所示。最后单击【OK】按钮即可。此时在 app/src/main/res 文件夹下会新建一个 values-zh 的子文件夹，将项目 App0201 中文件夹 app/src/main/res/values 中的文件 strings.xml 复制到 values-zh 的子文件夹，打开 strings.xml 并将相应的内容改成中文即可。

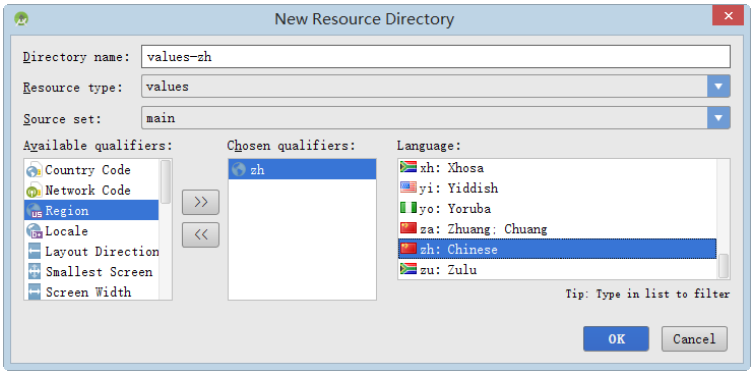


图 2-32　在【New Resource Directory】中选择“zh:Chinese”

【任务 2-4】 将 Android Studio 项目打包生成 APK

【任务描述】

创建的 Android Studio 项目代码编写完成后，需要在真机环境下测试其功能，这时需要将项目打包生成 APK，将 App0201 项目打包生成 APK。

【知识索引】

（1）APK

APK 是 Android Package 的缩写，即 Android 安装包。APK 是类似 Symbian Sis 或 Sisx 的文件格式。通过将 APK 文件直接传到 Android 模拟器或 Android 手机中执行即可安装。

（2）将 Android Studio 项目打包生成 APK。

【实施过程】

Android Studio 项目打包生成 APK 的具体步骤如下：

（1）在 Android Studio 主界面，选择菜单命令“Build”→“Generate Signed APK”，如图 2-33 所示，打开“Generate Signed APK Wizard”对话框，在对话框中单击【Create NEW…】按钮打开“New Key Store”对话框。

在该对话框中单击“Key store path”右侧的【…】按钮，打开【Choose keystore file】，在对话框中选择 APK 存放路径和文件名称，如图 2-34 所示。

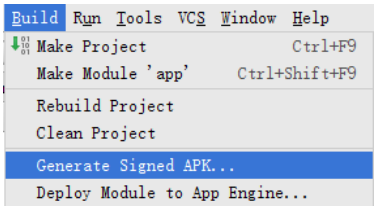


图 2-33　选择菜单命令“Build”→“Generate Signed APK”

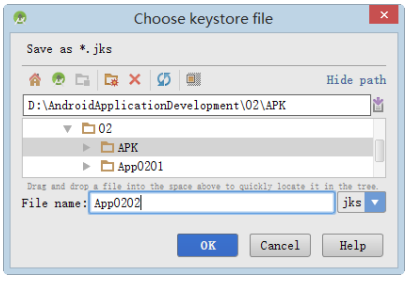


图 2-34　“Choose keystore file”对话框

然后单击【OK】按钮返回“New Key Store”对话框，在该对话框中输入必要的配置信息，例如密码、别名等，如图 2-35 所示。

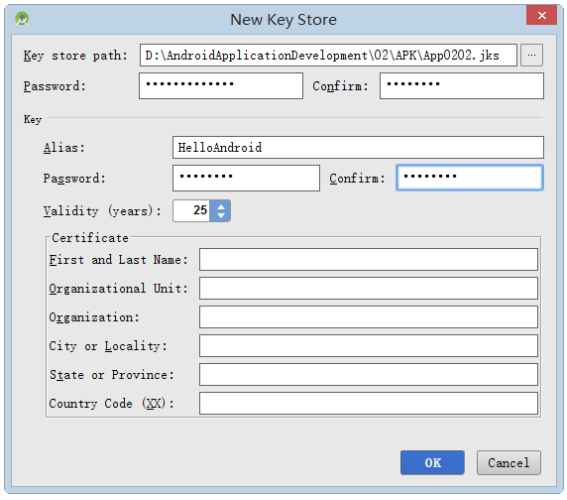


图 2-35　“New Key Store”对话框

单击【OK】按钮返回“Generate Signed APK Wizard”对话框，如图 2-36 所示。

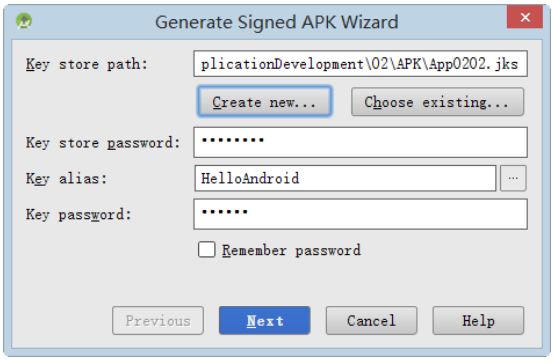


图 2-36　在“Generate Signed APK Wizard”对话框显示相关配置信息

配置完成后单击【Next】按钮进入下一步，如图 2-37 所示，单击【Finish】按钮，然后 Android Studio 开始生成 APK，完成后弹出如图 2-38 所示的“Generate signed APK”对话框，单击【Close】按钮即可。

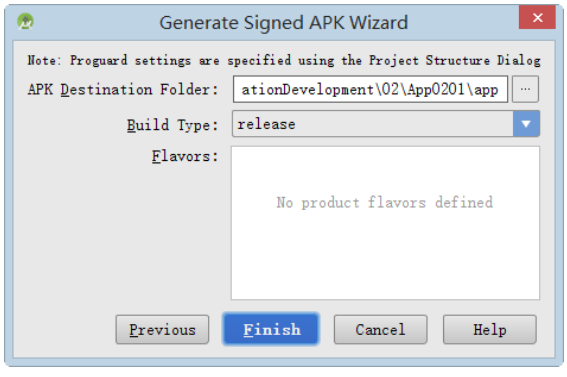


图 2-37　在“Generate Signed APK Wizard”对话框中单击【Finish】按钮

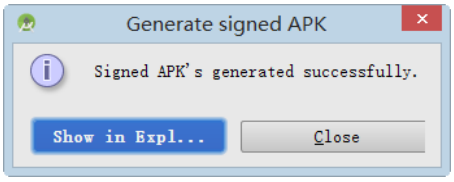


图 2-38 “Generate signed APK”对话框

单元小结

Android Studio 提供了集成的 IDE 用于开发和调试 Android 应用程序，是一款性能良好、开发效率高的 Android 应用开发工具。本单元主要认识了 Android 应用程序中的布局文件 activity_main.xml、MainActivity.java 文件和 AndroidManifest.xml 文件的基本组成及其含义，认识了 Android 应用程序的样式和主题设置，熟悉了 Android Studio 的组成结构，学会了 Android Studio 项目中的模块操作、开发环境的个性化设置和将 Android Studio 项目打包生成 APK 的方法。

单元习题

一、填空题

1．一个 Android 项目成功创建后，默认生成一个布局文件 activity_main.xml，该文件位于项目的（　　）文件夹中。

2．新建一个 Android Studio 项目时，会默认生成一个 Activity 文件 MainActivity.java，该文件中会自动导入 2 个类，分别是（　　）和（　　），同时导入 Menu 和 MenuItem 2 个接口。

3．在 Android 应用程序中，界面是通过布局文件设定的，布局文件采用（　　）格式。

4．在 Android 程序中，src 文件夹是用于放置程序的（　　）。

5．Android 应用程序的配置文件名称为（　　）。

6．Android 程序开发完成后，若要发布到 Internet 上供别人用，需要将程序（　　）。

二、选择题

1．如果需要创建一个字符串资源，需要将字符串放在 res\values 的哪个文件中？（　　）

A．value.xml　　B．strings.xml　　C．dimens.xml　　D．styles.xml

2．新建的 Android 程序默认的布局方式是相对布局（RelativeLayout），该布局中默认包含一个（　　）控件。

A．ImageView　　B．TextView　　C．EditText　　D．Button

3．要让布局文件或者控件能够显示在界面上，必须要设置 RelativeLayout 和控件的（　　）。

A．宽度或高度　　B．宽度和 id　　C．宽度和高度　　D．高度和 id

4．AndroidManifest.xml 配置文件中，activity 和 application 里都可以设置 android:label 标签，activity 的优先级（　　）application。

A．低于　　B．等于　　C．高于　　D．无法确定

5．Android 项目启动时最先加载的是 AndroidManifest.xml 文件，如果有多个 Activity，以下哪个属性决定了该 Activity 最先被加载？（　　）

A．android.intent.action.MAIN　　B．android.intent.action.LAUNCHER

C．android.intent.action.ACTIVITY　　D．android.intent.action.ICON

6．Android 安装软件的后缀名是（　　）。

A．.sis　　B．cab　　C．.apk　　D．.jar

三、简答题

1．如何区分 Android Studio 中的 Project 和 Module？

2．如何在 Android Studio 开发环境中打开 Android 项目文件所在的目录？

3．在 Android 的布局文件 activity_main.xml 中“@+id/username”与"@id/username"两者有何区别？

4．Android 程序如何获取界面上的控件并在窗口中显示？

5．简述 Android 项目中重要的文件夹和文件，它们的作用是什么？

单元 3

Android的界面设计与资源应用程序设计

Android 应用程序中的用户界面（UI）设计非常重要，它是人机之间数据传递、交互信息的重要媒介和对话接口，是 Android 程序的重要组成部分。Android 系统中提供了许多控件，包括文本控件、按钮控件、图片控件、时间控件等，控件可以为应用程序构建美观、易用的用户界面。常用的控件定义在 android.view.View 和 android.view.ViewGroup 两个类中，View 类是 Android 系统中构建用户界面的基本类，ViewGroup 是一个特殊的 View 类，它的功能就是装载和管理下一层的 View 对象和 ViewGroup 对象。本单元主要探讨使用控件和资源构建实用、美观的用户界面。



教学导航

教学目标	（1）理解 Android 屏幕元素的层次结构、View 与 ViewGroup （2）熟悉 Android 的布局对象和 Android 中创建 UI 界面的方式 （3）熟悉 Android 常用的 UI 控件及其基本属性 （4）熟悉 Android 尺寸的单位 （5）掌握 TextView、EditText、Button、DatePicker、TimePicker 等常用控件的功能、基本属性以及重要方法 （6）学会使用文字标签显示欢迎信息、设计包含多种控件的用户登录界面 （7）学会设计开关与调节声音的界面、使用 DatePicker 控件获取日期和使用日期时间控件获取与显示时间 （8）学会使用 LinearLayout 布局设计用户注册界面、使用 TableLayout 布局设计商品信息展示界面、使用 GridLayout 布局设计仿 Windows 8 桌面、使用 FrameLayout 布局实现图片相框效果
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	10 课时



知识导读

3.1 Android 屏幕元素的层次结构

一个 Android 应用程序最基本的功能单元是行为（Activity），即 android.app.Activity 类的一个对象。一个行为（Activity 实例）可以做很多事情，但是它本身并不能使自己显示在屏幕上，而是借助于 ViewGroup（视图组）和 View（视图），这两个才是 Android 平台上最基本的两个用户界面表达单元。

对于 Android 应用程序中的一个屏幕，它的屏幕元素是按层次结构来描述的。要将一个

屏幕元素层次树绑定在一个屏幕上显示，Activity 会调用它的 setContentView()方法并且传入这个层次树的根节点引用。当 Activity 被激活并且获得焦点时，系统会通知 activity 并且请求根节点去计算并绘制树，根节点就会请求它的子节点去绘制它们自己。

每个树上的 ViewGroup 节点会负责绘制它的子节点。ViewGroup 会计算它的有效空间，布局所有的子显示对象，并最终调用所有的子显示对象的 Draw()方法来绘制显示对象。各个子显示对象可以向父对象请求它们在布局中的大小和位置，但最终决定各个子显示对象的大小和位置的是父对象。

大多数的界面控件都在 android.view 和 android.widget 包中，android.view.View 是它们的父类。Android 的原生控件，一般是在 res/layout 下的 XML 文件中声明，然后在 Activity 文件中通过使用 super.setContentView(R.layout.某布局 layout 文件名)来加载 layout。在 Activity 类中获取控件的引用需要使用 super.findViewById(R.id.控件的 ID)，接着就可以使用这个引用对控件进行操作，例如添加监听，设置内容等。当然也可以通过代码动态的使用控件。

（1）View 子类结构

View 子类结构图如图 3-1 所示。

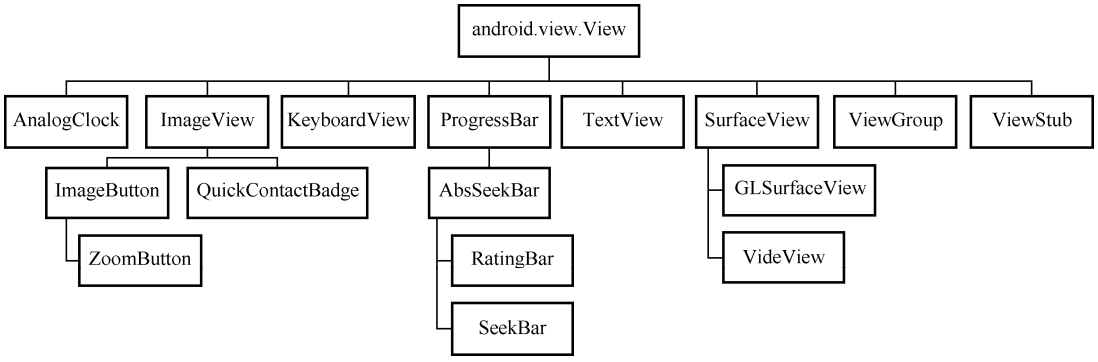


图 3-1　View 子类结构图

（2）ViewGroup 子类结构

ViewGroup 子类结构图如图 3-2 所示。

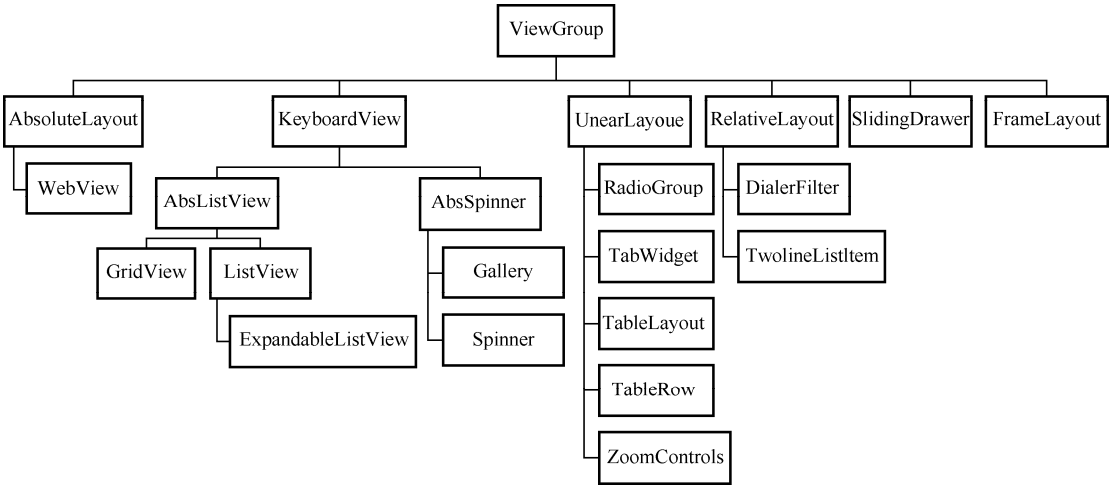


图 3-2　ViewGroup 子类结构图

（3）TextView 子类结构

TextView 子类结构如图 3-3 所示。

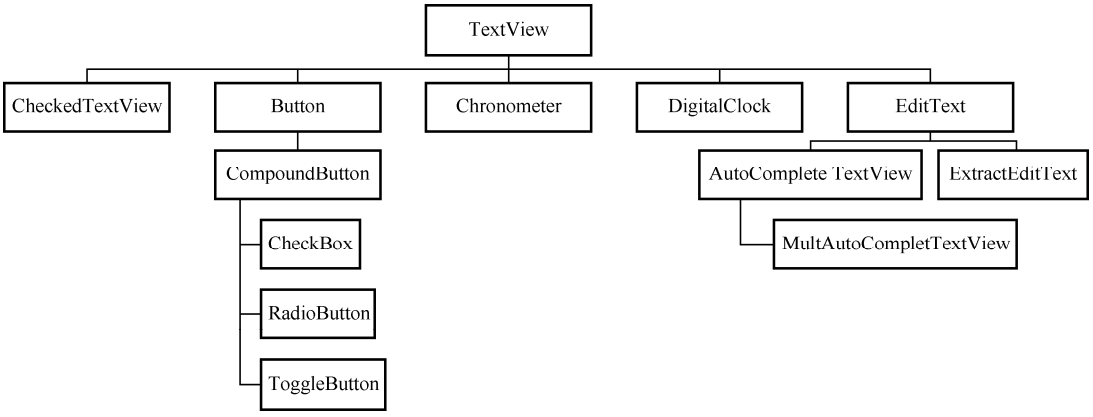


图 3-3　TextView 子类结构

（4）FrameLayout 子类结构

FrameLayout 子类结构如图 3-4 所示。

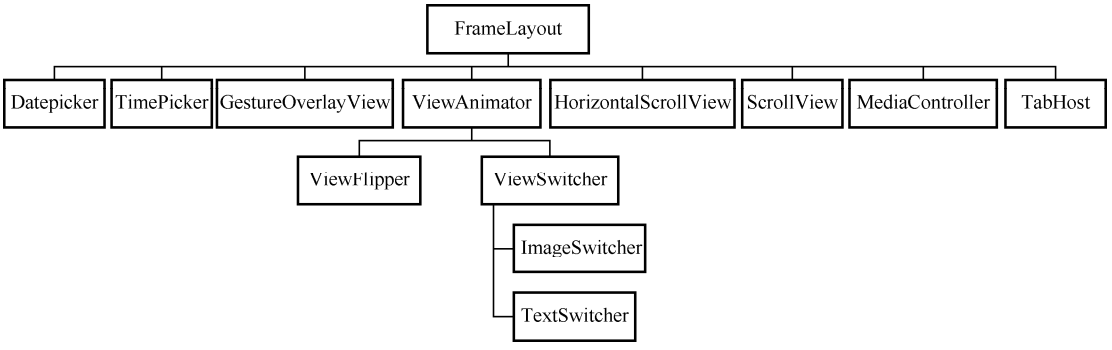


图 3-4　FrameLayout 子类结构

3.2 View 与 ViewGroup

在 Android SDK UI 的核心控制中有 android.view.View 和 android.view.ViewGroup 两个主要类。View 表示是一个视图，就是一般视觉上的一个区域，ViewGroup 也是一个 View，它扩展 View 使其内部能存放其他的 View，可以理解为一个 Container 容器。ViewGroup 内部采用跟 Swing 一样的处理机制，内部采用一个 layout manager 来管理它的布局，让用户能采用内置的布局进行视图控制。

在 Android APP 中，所有的用户界面元素都是由 View 和 ViewGroup 的对象构成的。View 是绘制在屏幕上的，用户能与之交互的一个对象，而 ViewGroup 则是一个用于存放其他 View 和 ViewGroup 对象的布局容器。

APP 的用户界面上的每一个组件都是使用 View 和 ViewGroup 对象的层次结构来构成的，如图 3-5 所示，每个 ViewGroup 都是要给看不见的用于组织子 View 的容器，而它的子 View 可能是输入控件或者在 UI 上绘制了某块区域的小部件。有了层次树，就可以根据自己的需要，

设计简单或者复杂的布局了。

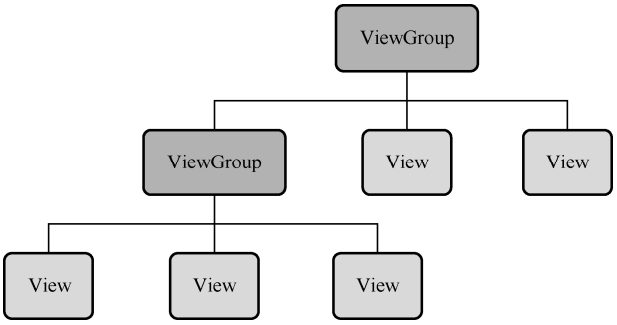


图 3-5　一个 UI 布局的层次结构示意图

无需全部用 View 和 ViewGroup 对象来创建 UI 布局，Android 给我们提供了一些 APP 控件和 UI 布局，只需要定义其内容即可。

（1）视图对象（View）

在 Android 程序中 View 类是最基本的一个 UI（User Interface，用户界面）类，基本上所有高级的 UI 控件都继承这个类。一个 View 通常占用屏幕上的一个矩形区域，并负责绘图及事件处理，并且可以设置该块区域是否可见，以及获取焦点等操作。View 是所有窗体部件的基类，是为窗体部件服务的，这里的窗体部件即 UI 控件，例如一个按钮或文本框。Android 提供了一系列的标准 UI 控件供我们直接使用，同时，我们也可以通过继承于 View 类或 View 的子类，来实现我们自定义的 UI 控件。

（2）视图容器对象（ViewGroup）

ViewGroup 是一个特殊的 View 类，它继承于 android.view.View，一个 ViewGroup 对象是一个 Android.view.ViewGroup 的实例，它的功能就是装载和管理下一层的 View 对象和 ViewGroup 对象，一个 ViewGroup 也可以加入到另一个 ViewGroup 当中。ViewGroup 是布局管理器（layout）及 view 容器的基类。ViewGroup 中，还定义了一个嵌套类 ViewGroup.LayoutParams。这个类定义了一个显示对象的位置、大小等属性，view 通过 LayoutParams 中的这些属性值来告诉父级容器，它们将如何放置。

继承于 ViewGroup 的一些主要的布局类有 RelativeLayout、LinearLayout、TableLayout、FrameLayout 和 AbsoluteLayout。

3.3 Android 的布局对象

Android 的布局管理控件用于把多个控件集成在一个用户界面中。Android 常见的布局对象有：RelativeLayout（相对布局）、LinearLayout（线性布局）、TableLayout（表格布局）、FrameLayout（帧布局）和 AbsoluteLayout（绝对布局），它们都继承了 ViewGroup，作为各种不同布局管理模型的容器，它们都提供了各自独到的功能。

（1）RelativeLayout 布局

RelativeLayout 是相对布局，允许通过指定显示对象相对于父容器或其他兄弟控件的相对位置结合 margin、padding 来进行布局。例如一个按钮可以放于另一个按钮的右边，或者可以放在布局管理器的中央。RelativeLayout 是一个比较复杂的布局管理对象，它是用定义各个视

图相对的位置来管理布局，它一个好用的地方是嵌套的布局层次少。

RelativeLayout 布局的属性说明：

① 相对于给定 ID 兄弟控件位置关系属性，属性的取值为相对引用控件的 ID

android:layout_above：将该控件的底部置于给定 ID 的控件的上方。

android:layout_below：将该控件的底部置于给定 ID 的控件的下方。

android:layout_toLeftOf：将该控件的右边缘与给定 ID 的控件左边缘对齐。

android:layout_toRightOf：将该控件的左边缘与给定 ID 的控件右边缘对齐。

android:layout_alignBaseline：将该控件的 baseline 与给定 ID 的控件基准线对齐。

android:layout_alignTop：将该控件的顶部边缘与给定 ID 的控件顶部边缘对齐。

android:layout_alignBottom：将该控件的底部边缘与给定 ID 的控件底部边缘对齐。

android:layout_alignLeft：将该控件的左边缘与给定 ID 的控件左边缘对齐。

android:layout_alignRight：将该控件的右边缘与给定 ID 的控件右边缘对齐。

android:layout_toLeftOf：将该控件的右边缘与给定 ID 的控件左边缘对齐。

② 相对于父容器的位置关系属性，属性的取值为 true 或 false

android:layout_alignParentTop：如果取值为 true，将该控件顶部与其父控件顶部对齐。

android:layout_alignParentBottom：如果取值为 true，将该控件底部与其父控件底部对齐。

android:layout_alignParentLeft：如果取值为 true，将该控件左部与其父控件左部对齐。

android:layout_alignParentRight：如果取值为 true，将该控件右部与其父控件右部对齐。

③ 居中位置设置，属性的取值为 true 或 false

android:layout_centerHorizontal：如果取值为 true，将该控件置于水平居中。

android:layout_centerVertical：如果取值为 true，将该控件置于垂直居中。

android:layout_centerInParent：如果取值为 true，将该控件置于父控件的中央。

④ 指定偏移（外边距）量的属性

android:margin：设置控件的偏移值

android:layout_marginTop：设置控件上边的偏移值。

android:layout_marginBottom：设置控件下边的偏移值。

android:layout_marginLeft：设置控件左边的偏移值。

android:layout_marginRight：设置控件右边的偏移值。

⑤ 指定填充（内边距）量的属性

android:padding：设置控件内部内容的填充值。

android: paddingTop：设置控件内部内容上边的填充值。

android: paddingBottom：设置控件内部内容下边的填充值。

android: paddingLeft：设置控件内部内容左边的填充值。

android: paddingRight：设置控件内部内容右边的填充值。

【注意】：区分 margin 和 padding 的不同，首先 margin 代表的是偏移，例如 marginleft= "5dp "，表示控件离容器左边缘偏移 5dp。而 padding 代表的则是填充，而填充的对象针对的是控件中的内容，例如 TextView 中的文字，例如为 TextView 设置 paddingleft ="5dp"，则是在控件里的内容的左边填充 5dp 的空间。margin 针对的是容器中的控件，而 padding 针对的是控件中的内容，要区分开来。父容器定位属性示意图如图 3-6 所示。

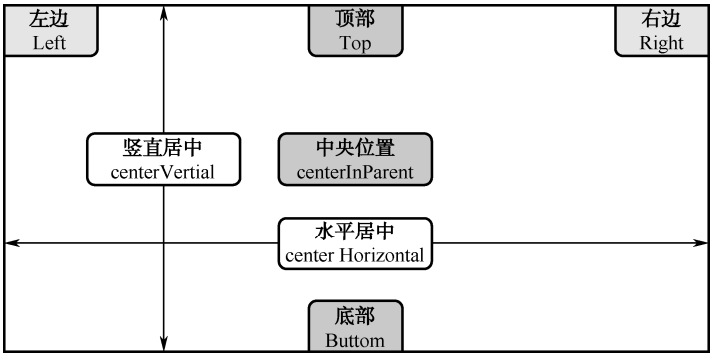


图 3-6　父容器定位属性示意图

⑥ 其他常用属性

android:grivity：指定控件内容文本的对齐位置，例如居中、居右等位置，指的是控件中的文本位置并不是控件本身。例如 android:gravity="left"表示控件中的文本居左显示。

ignoreGravity：设置了该属性为 true 的控件，将不受 android:grivity 属性设置的影响。

android:width：指定控件的宽度。

android:height：指定控件的高度。

android:sigleLine：如果设置为 true，则控件的内容在同一行中进行显示。

（2）LinearLayout 布局

LinearLayout 是一个最常用的基础布局对象，它以单一方向对其中的显示对象进行排列显示，如以垂直排列显示，则布局管理器中将只有一列；如以水平排列显示，则布局管理器中将只有一行。同时，它还可以对个别的显示对象设置显示比例。

LinearLayout 布局可以实现水平布局和垂直布局，它通过设置 android:orientation 属性将内部的所有子视图以横向或纵向进行排列。如果将布局方向设置为“vertical”，则表明垂直方向布局；设置为“horizontal”则表示水平方向布局。在 LinearLayout 中还可以设置内部子视图的方位(gravity)，Gravity 是一个排列属性，可以通过设置它的值为 left、center 或 right 而达到将视图放在父视图的左边、中间或右边的地方。

将多个横向的 LinearLayout 以当成子视图放进一个纵向的 LinearLayout 中，这样就形成一个类似 Table 的布局，不过如果是这种情况，最好是用 TableLayout 比较合适。

（3）TableLayout 布局

TableLayout 布局以拥有任意行列的表格对显示对象进行布局，每个显示对象被分配到各自的单元格之中，但单元格的边框线不可见。

TableLayout 布局管理对象会将容器内部的子节点按照行列进行排序，它就像是 html 中的<table>节点。在 TableLayout 节点内部中声明<TableRow>将为 table 添加一行，<TableRow>中的子视图将排成 *N* 列，如果<TableRow>内有 *N* 个子视图就将该行划分为 *N* 列。整个表的列是根据表格每行中最多有多少列来决定的，假设整个表格中有 2 行 3 列，其中有一行为 4 列则表格将分为 4 列。TableLayout 默认会将它的直接子节点当成一行，所以可以放置任意的 View 到 TableLayout 中，而且对于 TableLayout 的直接子节点设置 layout_width="wrap_content" 属性将不会生效，因为默认它会重写成 match_parent。

TableLayout 还有一个特殊的属性：stretchColumns 用来设置需要拉宽的列，因为有的列

可能不会填满区域，为了布局的美观可以定义哪个列将进行拉宽。用逗号隔开的列值来指定哪些列需要拉宽，列值是从 0 开始的，如果需要拉升第 2 和第 3 列，你可以设置 stretchColumns="1,2"。同样的道理，如果需要紧缩一个列可以使用 shrinkColumns 来指定，如果要让某些列不可见，可以通过设置 collapseColumns 属性，这些用法都跟 stretchColumns 一样。

（4）FrameLayout 布局

FrameLayout 是一个最简单的布局对象，它里面只显示一个显示对象。在屏幕上预留好一块空白的区域，所有的元素都被放置在 FrameLayout 区域的左上方，无法给这些元素一个确切的位置，如果有很多值元素则会重叠在前一个元素上，只有最上面的元素能被看到，如果设置它的背景是透明的，则可以透过上面的元素看到下面的元素。

（5）GridLayout 布局

GridLayout 布局样式和 LinearLayout 样式一样，有水平和垂直两个方向的布局方式。即如果设置为垂直方向布局，则下一个单元格将会在下一行的同一位置或靠右一点的位置出现，而水平方向的布局，则意味着下一个单元格将会在当前单元格的右边出现，也有可能会跨越下一行，其原因是在 GridLayout 在布置中，定义每个子控件与以前使用布局中定义的方法有点不同，默认的是对所有的子控件使用 wrap_content 的方式，而不是显式声明宽度和高度或使用 wrap_conent 和 match_parent。

（6）AbsoluteLayout 布局

AbsoluteLayout 布局允许以坐标的方式，指定显示对象的具体位置，左上角的坐标原点为 (0, 0)，其中第 1 个 0 代表 x 坐标，第 2 个 0 代表 y 坐标，向下及向右，坐标值变大。在 AbsoluteLayout 布局中子元素可以相互重叠，各个元素之间没有任何联系，控件可以在任意位置摆放，元素的位置通过 android:layout_x 和 android:layout_y 属性决定，如果元素没有设置位置属性，则表示处于坐标原点位置，即屏幕的左上角位置。

（7）ScrollView 布局

由于移动终端的屏幕尺寸有限，经常会遇到在一个屏幕中无法将所有的控件或信息显示完整的情况，为了解决这样的问题，通常需要应用滚动布局方式来实现。

ScrollView 的父类是 FrameLayout，它拥有 FrameLayout 的特性，另外当 ScrollView 中拥有很多内容，屏幕无法显示完整时，会通过滚动条进行显示。ScrollView 只支持垂直滚动。

3.4 Android 中创建 UI 界面的方式

在 Android 中有三种方式可以用来创建 UI 界面。

（1）在 XML 布局文件中创建 UI 视图的层次架构

Android 提供了很多属性用来使用 XML 对视图和布局进行管理，可以通过这些属性定义视图大小，方向等。使用 XML 布局创建 UI 界面的优点是：使用 MVC 模式将视图层和逻辑控制层分开。视图和布局控制文件没有混合在程序代码中，使整个项目结构清晰，可维护性高。在 XML 布局文件中创建 UI 界面的 XML 代码（即 activity_main.xml 文件对应的 XML 代码）如表 3-1 所示。

表 3-1　在 XML 布局文件中创建 UI 界面的 XML 代码

序号	布 局 代 码
01	<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	android:orientation="vertical"
03	android:layout_width="match_parent"
04	android:layout_height="match_parent" >
05	<LinearLayout
06	android:orientation="horizontal"
07	android:layout_width="match_parent"
08	android:layout_height="wrap_content" >
09	<TextView
10	android:layout_width="wrap_content"
11	android:layout_height="wrap_content"
12	android:text="　姓名：" />
13	<TextView
14	android:id="@+id/tvName"
15	android:layout_width="wrap_content"
16	android:layout_height="wrap_content"
17	android:text="简单" />
18	</LinearLayout>
19	<LinearLayout
20	android:orientation="horizontal"
21	android:layout_width="match_parent"
22	android:layout_height="wrap_content" >
23	<TextView
24	android:layout_width="wrap_content"
25	android:layout_height="wrap_content"
26	android:text="　地址：" />
27	<TextView
28	android:id="@+id/tvAddress"
29	android:layout_width="wrap_content"
30	android:layout_height="wrap_content"
31	android:text="长沙" />
32	</LinearLayout>
33	</LinearLayout>

【注意】：每个 View 的 android:layout_width、android:layout_height 必须在 XML 文件中进行设置，否则程序将发生异常而终止。

对应 Activity 代码如下所示。

```
package com.example.exampleapp0301;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



图 3-7　项目 ExampleApp0301 的运行结果

项目 ExampleApp0301 的运行结果如图 3-7 所示。

（2）在 Activity 类中编写代码创建视图或布局

动态地创建 View 或 ViewGroup 并设置它们的属性，使用代码创建视图或布局的优点是：可以根据程序运行状态，动态地显示内容。在代码中创建视图或布局的示例代码（即 MainActivity.java 文件对应的代码）如表 3-2 所示。

表 3-2　在代码中创建视图或布局的示例代码

序号	布 局 代 码
01	package com.example.exampleapp0302;
02	import android.app.Activity;
03	import android.os.Bundle;
04	import android.view.Menu;
05	import android.view.MenuItem;
06	import android.view.ViewGroup.LayoutParams;
07	import android.widget.LinearLayout;
08	import android.widget.TextView;
09	public class MainActivity extends Activity {
10	@Override
11	protected void onCreate(Bundle savedInstanceState) {
12	super.onCreate(savedInstanceState);
13	setContentView(R.layout.activity_main);
14	//创建一个主视图容器
15	LinearLayout parent = new LinearLayout(this);
16	parent.setOrientation(LinearLayout.VERTICAL);
17	parent.setLayoutParams(new LayoutParams(
18	LayoutParams.MATCH_PARENT,LayoutParams.MATCH_PARENT));
19	//创建存放姓名的容器
20	LinearLayout nameLinearLayout = new LinearLayout(this);
21	nameLinearLayout.setOrientation(LinearLayout.HORIZONTAL);
22	nameLinearLayout.setLayoutParams(new LayoutParams(
23	LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
24	TextView nameLabel = new TextView(this);
25	nameLabel.setText("　姓名：");
26	TextView name = new TextView(this);
27	name.setText("简单");
28	nameLinearLayout.addView(nameLabel);

续表

序号	布 局 代 码
29	nameLinearLayout.addView(name);
30	//创建存放地址的容器
31	LinearLayout addressLinearLayout = new LinearLayout(this);
32	addressLinearLayout.setOrientation(LinearLayout.HORIZONTAL);
33	addressLinearLayout.setLayoutParams(new LayoutParams(
34	LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
35	TextView addressLabel = new TextView(this);
36	addressLabel.setText("　地址：");
37	TextView address = new TextView(this);
38	address.setText("长沙");
39	addressLinearLayout.addView(addressLabel);
40	addressLinearLayout.addView(address);
41	//将姓名、地址放到主容器中
42	parent.addView(nameLinearLayout);
43	parent.addView(addressLinearLayout);
44	//显示主容器
45	setContentView(parent);
46	}
47	}

表 3-2 所示的代码中，主容器、姓名容器和地址容器都是 LinearLayout 布局，它有一个属性可以设置方向，在主容器中方向是纵向的，而其他 2 个则是横向的，所以姓名标签和姓名值在同一行，地址标签和地址值在同一行。因为主容器是纵向的，所以主容器里面的 2 个子节点姓名容器和地址容器将从上到下排列，所以分成 2 行。项目 ExampleApp0302 的运行结果与项目 ExampleApp0301 的运行结果相同，如图 3-7 所示。

（3）混合使用动态创建和 XML 布局创建视图或布局

在 XML 中声明视图或布局，然后在代码中找出这些视图，修改里面的属性。activity_确定 main.xml 文件对应的 XML 代码如表 3-1 所示，MainActivity.java 文件对应的代码如表 3-3 所示。

表 3-3　MainActivity.java 文件对应的代码

序号	布 局 代 码
01	package com.example.exampleapp0303;
02	import android.app.Activity;
03	import android.os.Bundle;
04	import android.view.Menu;
05	import android.view.MenuItem;
06	import android.widget.TextView;
07	public class MainActivity extends Activity {
08	@Override
09	protected void onCreate(Bundle savedInstanceState) {
10	super.onCreate(savedInstanceState);

续表

序号	布 局 代 码
11	setContentView(R.layout.activity_main);
12	TextView name = (TextView)findViewById(R.id.tvName);
13	name.setText("高兴");
14	TextView address = (TextView)findViewById(R.id.tvAddress);
15	address.setText("深圳");
16	}
17	}

项目 ExampleApp0303 先将布局文件 activity_main.xml 设置到当前的布局中。然后根据 id 进行查找，查找出在 XML 布局中声明的 TextView 对象，然后修改对应控件的值。

项目 ExampleApp0303 的运行结果如图 3-8 所示。



图 3-8　项目 ExampleApp0303 的运行结果

从图 3-8 所示的运行结果可以看出，原来在 activity_main.xml 布局文件中声明的内容被动态修改了。

通过以上三个创建 UI 界面方式对应的代码对比可以发现，使用 XML 布局文件的效果更好，因为避免写了一大堆界面代码嵌套在应用程序逻辑实现代码中。当布局复杂时还有很多布局额外的属性要配置，例如 layout_padding、layout_margin 等，如果把这些东西全部写在代码中，那代码将会变得很臃肿，难以维护，所以除非要创建的只有简单的几个视图对象，否则最好用 XML 布局或混合方式。

3.5 Android 常用 UI 控件简介

控件是对数据和方法的封装，控件可以有自己的属性和方法。Android 内建了一个内部的 UI 框架，并做了一些特殊的设计以让其在手机上能有良好的用户体验。在 Android SDK 中，跟 JDK 一样有 Button、TextField、List、Grid 等，但是这些控件都做了优化，提供了适合在手机上做的控制。Android 提供的 UI 控件允许建立应用程序的图形用户界面。

设计 Android 应用程序用户界面时常用的 UI 控件如下：

（1）TextView 控件

TextView 用于显示文本给用户。

TextView 用于显示一些不能编辑的文本信息，看起来它就像是 GUI（Graphical User Interface，简称 GUI，又称图形用户接口）里常见的 Label，但是其实它不止是 Label 那么简单。它还提供一些独特的控制功能，例如可以将 TextView 里面的网址和 Email 链接转换成链接形式，在 XML 布局中使用属性 android:autoLink 即可实现。

TextView 控件的示例代码如下所示：

```
<TextView
    android:id="@+id/textview1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoLink="all"
    android:text="欢迎访问 www.baidu.com"  />
```

（2）EditText 控件

EditText 是 TextView 预定义的子类，包括丰富的编辑功能。

EditText 是 TextView 的子类，EditText 扩展了 TextView 使其具备能编辑文字的能力。可以设置 EditText 的很多控制功能，例如首字母大写、只允许输入数字、当成密码输入框等，还可以引用一些有定义样式的文本内容。

EditText 控件的示例代码如下所示：

```
<EditText
    android:id="@+id/edittext1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/text1"  />
```

（3）AutoCompleteTextView 控件

AutoCompleteTextView 继承了 EditText，并且添加了额外的 AutoComplete 功能，能够实现动态匹配输入的内容。用户需要先设置一个 Adapter 以提供 AutoComplete 所需的数据源（AutoComplete 就像是使用 Google 或百度搜索时提示的建议搜索词）。它实际上由 2 部分组成：一个是 EditText 用来输入字符，另一个是建议列表的 List。用户需要在代码中指定如何显示 AutoComplete 的建议列表。在用户输入时，它会显示自动完成建议的列表。

AutoCompleteTextView 控件常用的方法如下：

① clearListSelection()：用于清除选中的列表项。

② dismissDropDown()：用于关闭存在的下拉菜单。

③ getAdapter()：用于获取适配器。

（4）MultiAutoCompleteTextView 控件

MultiAutoCompleteTextView 继承了 AutoCompleteTextView，并且扩展了它的功能，能够对用户键入的文本进行有效的扩充提示，而不需要用户输入整个内容。因为在使用 AutoCompleteTextView 时有一个不便的地方是：它做的是整个 TextView 中文本信息的全匹配。假设要输入的是一个句子而又要具备能提示里面单词的功能的话，AutoCompleteTextView 就完成不了，因为它会将整个句子都作为匹配对象所以无法进行单词建议，而 MultiAutoCompleteTextView 则解决了这个问题，它会根据用户所打的字进行匹配而不是整个 TextView 中的文本信息，它的使用方法跟 AutoCompleteTextView 差不多。

（5）Button 控件

按钮在 Android 中是 android.wiget.Button 类，可以供用户进行单击操作，按钮上通常显示文字，也可以通过 android:background 属性为 Button 设置背景色或背景图片。它继承了 TextView，所有 TextView 可配置的属性都可以在 Button 上配置，所以我们可以定义很多 Button 的样式，同时 Button 还扩展了部分功能让用户可以在文字的上下左右四个方向添加 icon 图片。

Button 的示例代码如下。

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="普通按钮"
    android:typeface="serif"
    android:textStyle="bold"  />
<Button
```

```
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="带图片按钮"
        android:drawableTop="@drawable/icon"  />
```

（6）ImageView 控件

ImageView 控件用于在屏幕上显示任何 Drawable 对象，通常用来显示图片，ImageView 继承自 view 类。ImageView 控件的主要属性如表 3-4 所示。

表 3-4　ImageView 控件的主要属性

属 性 名 称	描　　述
android:adjustViewBounds	设置 ImageView 是否调整自己的边界来保持所显示图片的宽高比，需要与 maxWidth、MaxHeight 一起使用，否则单独使用没有效果
android:cropToPadding	设置是否截取指定区域用空白代替，单独设置无效果，需要与 scrollY 一起使用
android:maxHeight	设置 View 的最大高度，单独使用无效，需要与 setAdjustViewBounds 一起使用，将 adjustViewBounds 设置为 true。如果想设置图片固定大小，又想保持图片宽高比，需要进行如下设置： ① 设置 setAdjustViewBounds 为 true； ② 设置 maxWidth、MaxHeight 属性值； ③ 设置 layout_width 和 layout_height 为 wrap_content
android:maxWidth	设置 View 的最大宽度，使用方法同 android:maxHeight
android:scaleType	设置显示的图片如何缩放或者移动以适应 ImageView 控件的大小，其取值如下： ① fitXY：对图像的横向与纵向进行独立缩放，使得该图片完全适应 ImageView 尺寸，但是图片的横纵比可能会发生改变 ② fitStart：保持纵横比缩放图片，缩放完成后图片的高度与 ImageView 控件高度相同，且将图片放于 ImageView 的左上角 ③ fitCenter：保持纵横比缩放图片，缩放后放于 ImageView 的中间 ④ fitEnd：保持纵横比缩放图片，缩放后放于 ImageView 的右下角 ⑤ center：直接让图片按原图大小在 ImageView 中居中显示，但图片宽高大于 ImageView 的宽高时，截图显示图片中间部分 ⑥ centerCrop：按比例放大原图直至等于 ImageView 某边的宽高显示，可能会出现图片显示不完全的现象 ⑦ centerInside：当原图宽高或等于 ImageView 的宽高时，按原图大小居中显示；反之将原图缩放至 ImageView 的宽高居中显示 ⑧ matrix：使用 matrix 方式进行缩放显示
android:src	设置 ImageView 控件要显示的图片，例如，android:src="@drawable/ btn_bg 表示该控件要显示 rew\drawable 文件夹下 btn_bg 这张图片
android:tint	将图片渲染成指定的颜色

ImageView 控件设置其要显示图片的方法有 setImageResource()和 setImageBitmap()。

ImageView 控件的示例代码如下所示。

```
 <ImageView
    android:src="@drawable/ btn_bg "
    android:id="@+id/imageView1"
```

```
    android:maxWidth="200px"
    android:maxHeight="200px"
    android:adjustViewBounds="true"
    android:layout_margin="5px"
    android:layout_height="180px"
    android:layout_width="180px"
    android:scaleType="centerInside"
    android:background="@android:color/white"
    android:tint="#ffff00"  />
```

（7）ImageButton 控件

ImageButton 按钮的内容是图片，它与 Button 不一样，Button 的内容可以是文字或者文字+图片，但是 ImageButton 的内容只能是图片，而且 ImageButton 是继承 ImageView，而不是 Button，所以二者是不一样的。可以通过在 XML 布局里设置 android:src 属性指定显示的图片地址，也可以动态地通过代码调用 setImageResource()方法来填充一个图片。

ImageButton 示例代码如下所示。

```
<ImageButton
    android:id="@+id/ibtnLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon" />
```

（8）ToggleButton 控件

ToggleButton 和 Checkbox、radio 按钮一样是一个有状态的按钮，它有一个开启或关闭状态，在开启状态时，按钮的下面有一条绿色的粗线，当它是在关闭状态时则粗线变成灰色的。可以通过配置 android:textOn 和 android:textOff 来对应两种不同状态时要显示的文字。

ToggleButton 示例代码如下所示。

```
<ToggleButton
    android:id="@+id/tbtnPlay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="播放"
    android:textOff="停止"  />
```

（9）CheckBox 控件

CheckBox 是一个复选框，它存在两种状态：选中和未选中。用户可以通过调用 CheckBox 对象的 setChecked()或 toggle()方法来改变复选框的状态，通过调用 isChecked 方法获取选中的值。可以通过调用 CheckBox 的 setOnCheckedChangeListener()方法来监听 Checkbox 的状态改变事件。

CheckBox 示例代码如下所示。

```
<CheckBox
    android:id="@+id/checkbox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Java"  />
<CheckBox
    android:id="@+id/checkbox2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="C#"
    android:checked="true"  />
```

（10）RadioButton 控件与 RadioGroup 控件

RadioButton 是一个单选框，只允许用户在多个选项里选择其中一个，RadioGroup 用于组织一个或多个单选按钮。为了让多个选项中仅能选择一个，我们需要将这多个选项放到一个分组中，这样默认每个分组只允许用户选中分组中的一项。为了要使用 RadioButton，首先我们要创建一个 RadioGroup 分组，然后将 RadioButton 放到分组中。所有在 RadioGroup 中的 RadioButton 默认都是未选中状态，当然也可以通过 XML 布局，设置某个 RadioButton 处于选中状态。跟 CheckBox 一样，可以调用 setChecked()和 toggle()方法来改变它里面的状态值，也可以监听它的状态改变事件。

RadioButton 示例代码如下所示。

```
<RadioButton
    android:id="@+id/radiobutton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="男"  />
<RadioButton
    android:id="@+id/radiobutton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="女"
    android:checked="true"  />
```

（11）DatePicker 控件与 TimePicker 控件

DatePicker 控件用来选取日期，TimePicker 控件用来选取一天中的时间。选择日期和时间，在很多应用程序中经常用到，Android 由于受到手机屏幕大小的限制无法像其他 GUI 框架一样创建复杂的布局来显示日期和时间，所以它们在 Android 中的展现比较简洁。我们可以在 XML 布局文件中通过声明 DatePicker 和 TimePicker 节点来创建这两个对象。

（12）ProgressBar 控件

ProgressBar 控件显示为进度条，也就是一个表示运转的过程，例如发送短信，连接网络等，表示一个过程正在执行中。一般只要在 XML 布局中定义就可以了。

ProgressBar 控件的常用方法如下：

① setIndeterminate()：设置进度条是否自动运转。

② setProgressStyle()：设置显示进度条风格，其取值为 ProgressDialog.STYLE_HORIZONTAL 或 ProgressDialog.STYLE_SPINNER。

③ setProgress()：设置进度条的进度。

④ setMax()：设置进度条的最大值。

⑤ getProgress()：获取进度条的当前进度。

ProgressBar 控件的示例代码如下：

```
<progressBar android:id="@+id/widget43"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical">
</ProgressBar>
```

此时，没有设置它的风格，那么它就是圆形的，一直会旋转的进度条。

添加以下代码，设置一个 style 风格属性，该 ProgressBar 就有了一个风格，这里表示大号

ProgressBar 的风格：

```
style="android:attr/progressBarStyleLarge"
```

（13）SeekBar 控件

SeekBar 是一个可拖动的进度条控件。拖动条类似进度条，不同的是用户可以拖动滑块改变 SeekBar 的值，例如手机的音量调节，同时还允许用户改变滑块的外观。由于拖动条可以被用户控制，所以需要对其进行事件监听，这就需要实现 SeekBar.OnSeekBarChangeListener 接口，监听滑块位置的改变。在 SeekBar 中需要监听 3 个事件，分别是：数值的改变（onProgressChanged）、开始拖动（onStartTrackingTouch）、停止拖动（onStopTrackingTouch）。在 onProgressChanged 中我们可以得到当前数值的大小。

SeekBar 控件的常用方法如下：

① getProgress()：获取拖动条的当前值。

② setMax()：设置拖动条的最大值。

③ setProgress：设置拖动条的当前值。

SeekBar 控件的示例代码如下：

```
<SeekBar
    android:id="@+id/seek"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="100"
    android:progress="50"
    android:secondaryProgress="75"   />
```

（14）Spinner 控件

Spinner 是一个下拉列表，允许用户从一组选择一个值（类似 HTML 中的 select）。

Spinner 控件常用的属性如下：

① dropDownHorizontal/VerticalOffset：用于设置下拉式列表框的水平/竖直偏移距离。

② dropDownWidth：用于设置下拉列表框的宽度。

③ popupBackground：用于设置列表框的背景。

④ prompt：用于设置列表框的提示信息（标题）。

（15）ListView 控件和 Grid 控件

列表多值显示是 Android 中一个常用的功能，如显示所有联系人、所有信息、文件目录等就需要列表显示控制。列表显示可以分为：纵向列表显示、Grid 网格排列、Gallery 横向显示。

ListView 控件主要用来显示纵向的列表。一般我们通过编写一个继承 ListActivity 的 Activity 类来做展示界面，因为 ListActivity 内置了一个 ListView 对象。通过调用 ListActivity.setListAdpater 将需要显示的数据使用一个 Adapter 绑定到 ListView 中。

ListView 是一个 ViewGroup，表明它是一个容器，里面可以放 View 对象，这边的 View 对象就是 ListView 里面的一条数据。

很多 UI 界面都需要使用 Grid 控件来显示二维数据，Grid 就像是 HTML 中的 table 用来显示一个表的行列值，Grid 也可以被用来显示图标，Android 系统启动后的菜单列出了所有应用就是用 Grid 控制的。

3.6 Android 尺寸的单位

Android 尺寸的常用单位如下：

（1）px（pixel）：是屏幕的像素点。

（2）in（inches）：英寸。

（3）mm（millimeters）：毫米。

（4）pt（points）：磅，1/72 英寸。

（5）dp：一个基于 density 的抽象单位，在每英寸 160 点的屏幕，1dp=1px。

（6）dip（device independent pixels）：等同于 dp。

（7）sp（scaled pixels-best for text size）：同 dp 相似，但还会根据用户的字体大小来缩放。

为了使界面能够在现在和将来的显示器类型上正常显示，一般建议始终使用 sp 作为文字大小的单位，将 dip/dp 作为其他元素的单位。另外也可以考虑使用矢量图，而不是位图。

3.7 Android 控件的基本属性

Android 控件的基本属性如下所示。

（1）android:id：为控件指定相应的 ID。

（2）android:text：指定控件当中显示的文字，需要注意的是，这里尽量使用 strings.xml 文件当中的字符串。

（3）android:gravity：指定 View 组件的对齐方式，例如居中，居右等位置，这里指的是控件中的文本相对于控件的位置并不是控件本身。

（4）android:layout_gravity：指定 Container 组件的对齐方式，例如一个 button 控件在 linearlayout 里，想把该 button 放在靠左、靠右等位置就可以通过该属性设置，以 button 为例，android:layout_gravity="right"则 button 靠右。

（5）android:textSize：指定控件当中字体的大小。

（6）android:background：指定该控件所使用的背景色，颜色采用 RGB 命名法。

（7）android:width：指定控件的宽度。

（8）android:height：指定控件的高度。

（9）android:layout_width：指定 Container 组件的宽度。

（10）android:layout_height：指定 Container 组件的高度。

（11）android:layout_weight：View 中很重要的属性，按比例划分空间。

（12）android:padding*：指定控件的内边距，也就是说控件当中的内容。

（13）android:sigleLine：如果设置为 true 的话，则控件的内容在同一行中进行显示。

（14）android:scaleType：是控制图片如何 resized 或 moved 来匹配 ImageView 的 siz。

（15）android:layout_centerHrizontal：水平居中。

（16）android:layout_centerVertical：垂直居中。

（17）android:layout_centerInparent：相对于父元素完全居中。

（18）android:layout_alignParentBottom：贴紧父元素的下边缘。

（19）android:layout_alignParentLeft：贴紧父元素的左边缘。

（20）android:layout_alignParentRight：贴紧父元素的右边缘。

（21）android:layout_alignParentTop：贴紧父元素的上边缘。

（22）android:layout_alignWithParentIfMissing：如果对应的兄弟元素找不到的话就以父元素做参照物。

（23）android:layout_below：在某元素的下方。

（24）android:layout_above：在某元素的上方。

（25）android:layout_toLeftOf：在某元素的左边。

（26）android:layout_toRightOf：在某元素的右边。

（27）android:layout_alignTop：本元素的上边缘和某元素的上边缘对齐。

（28）android:layout_alignLeft：本元素的左边缘和某元素的左边缘对齐。

（29）android:layout_alignBottom：本元素的下边缘和某元素的下边缘对齐。

（30）android:layout_alignRight：本元素的右边缘和某元素的右边缘对齐。

（31）android:layout_marginBottom：离某元素底边缘的距离。

（32）android:layout_marginLeft：离某元素左边缘的距离。

（33）android:layout_marginRight：离某元素右边缘的距离。

（34）android:layout_marginTop：离某元素上边缘的距离。

（35）android:paddingLeft：本元素内容离本元素右边缘的距离。

（36）android:paddingRight：本元素内容离本元素上边缘的距离。

（37）android:hint：设置 EditText 为空时输入框内的提示信息。

（38）android:LinearLayout：它确定了 LinearLayout 的方向，其值可以为 vertical（表示垂直布局）或 horizontal（表示水平布局）。

（39）android:editable="false"：设置 EditText 不可编辑。

（40）android:singleLine="true"：强制输入的内容为单行显示。

（41）android:ellipsize="end"：自动隐藏尾部溢出数据，一般用于文字内容过长一行无法全部显示的情况。

（42）android:interpolator：interpolator 定义一个动画的变化率（the rate of change），动画的进度使用 Interpolator 控制，可以实现匀速、正加速、负加速、无规则变加速等。Interpolator 是基类，封装了所有 Interpolator 的共同方法，它只有一个方法，即 getInterpolation (float input)。Android 提供了几个 Interpolator 子类，实现了不同的速度曲线。

Android 控件属性取值常见有以下类型：

（1）取值为 true 或 falser，例如 android:layout_centerHrizontal、android:layout_centerVertical、android:layout_alignParentLeft 等。

（2）取值必须为 id 的引用名"@id/id-name"，例如 android:layout_below、android:layout_above、android:layout_toLeftOf、android:layout_alignTop 等。

（3）取值为具体的像素值（如 30dip，40px），例如 android:layout_marginBottom、android:layout_marginLeft、android:layout_marginRight、android:layout_marginTop 等。

3.8 TextView 控件与 EditText 控件

1．功能说明

TextView 控件是文本表示控件，主要功能是向用户展示文本的内容，它是不可编辑的。

EditText 控件是编辑文本控件，主要功能是用于用户输入文本内容，它是具有编辑功能的 TextView。EditView 类是 TextView 类的子类，EditView 与 TextView 最大的不同就是用户可以对 EditView 控件进行编辑，同时还可以为 EditView 控件设置监听器，用来判断用户的输入是否合法。每一个控件都有着与之相应的属性，通过选择不同的属性，给予其值，能够实现不同的效果。

在程序设计和编写过程中，可以采用两种方式使用 TextView 和 EditText 控件。

方式一，通过在程序中创建控件的对象以来使用控件。例如 TextView 控件，可以通过编写如下代码完成控件使用。

```
TextView tv=new TextView(this);
tv.setText("用户名");
setContentView(tv);
```

方式二，通过在 res/layout 文件下的 XML 文件中布局使用控件。例如 TextView 控件，可以编写如下代码达到使用目的。

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="用户名"/>
```

相比而言，采用方式二更好，一是方便代码的维护，二是编码的灵活，三是利于分工协作。

2．基本属性

TextView 控件的基本属性如表 3-5 所示，这些属性大部分可用于 TextView，也可适用于 EditText 和 Button，仅有少部分属性只适用于其中之一。

表 3-5　TextView 控件的基本属性

属　性	描　述
android:autoLink	设置是否将文本显示为超链接形式，可选值为 none、web、email、phone、map、all。其中 none 表示所有文字显示为普通文本形式，没有超链接功能；web 表示网站 URL 链接会显示为超链接的形式，单击可以浏览网页；eamil 表示 E-Mail 地址会显示为超链接形式，单击可以发送邮件；phone 表示电话号码显示为超链接形式，单击可以拨号；map 表示地图地址显示为超链接形式；all 表示各类文字内容都显示为超链接形式
android:autoText	如果设置，将自动执行输入值的拼写纠正
android:background	设置的背景颜色或图片
android:bufferType	指定 getText()方式取得的文本类别，选项 editable 类似于 StringBuilder 可追加字符，也就是说 getText 后可调用 append 方法设置文本内容
android:capitalize	设置文本内容中的英文字母为大写，例如 android:capitalize = "characters"
android:cursorVisible	设定光标为显示或隐藏，默认状态为显示，取值为 false
android:digits	设置允许输入哪些字符，例如"1234567890.+-*/% ()"等
android:drawableBottom	在 text 的下方输出一个 drawable，例如图片。如果指定一个颜色的话会把 text 的背景设为该颜色，并且同时和 background 使用时覆盖后者
android:drawableLeft	在 text 的左边输出一个 drawable，例如图片
android:drawableRight	在 text 的右边输出一个 drawable，例如图片

续表

属　　性	描　　述
android:drawableTop	在 text 的正上方输出一个 drawable，例如图片
android:drawablePadding	设置 text 与 drawable（例如图片）的间隔，与 drawableLeft、 drawableRight、drawableTop、drawableBottom 一起使用，可设置为负数，单独使用没有效果
android:editable	设置控件的文本内容是否允许编辑
android:editorExtras	设置文本的额外输入数据
android:ellipsize	设置当文字过长，该控件的文本内容该如何显示。可选项包括"start"表示省略号显示在开头；"end"表示省略号显示在结尾；"middle"表示省略号显示在中间；"marquee"表示以跑马灯的方式显示（动画横向移动）
android:fontFamily	设置字体系列（由字符串命名）的文本
android:freezesText	设置保存文本的内容以及光标的位置
android:gravity	设置控件内文本的显示位置，如设置成"center"，文本将居中显示
android:hint	设置 Text 为空时显示的文字提示信息，可通过 textColorHint 设置提示信息的颜色，此属性在 EditView 中使用
android:id	定义控件的唯一标识 ID
android:includeFontPadding	设置文本是否包含顶部和底部额外空白，默认为 true
android:inputMethod	为文本内容的输入指定输入法，需要完全限定名(完整的包名)，例如：com.google.android.inputmethod.pinyin
android:inputType	设置文本的输入类型，用于帮助输入数据时显示合适的键盘类型，其可选项有 none（无特别限定）、text（输入普通字符）、textUri（URI 格式）、textEmailAddress（电了邮件地址格式）、textPassword（密码格式）、number（数字格式）、numberSigned（有符号数字格式）、numberDecimal（带小数点的浮动格式）、phone（拨号键盘）、datetime（日期时间键盘）、date（日期键盘）和 time（日间键盘）
android:ems	设置 TextView 的宽度为 N 个字符的宽度
android:maxEms	设置 TextView 的宽度为最长为 N 个字符的宽度，与 ems 同时使用时覆盖 ems 选项
android:minEms	设置 TextView 的宽度为最短为 N 个字符的宽度，与 ems 同时使用时覆盖 ems 选项
android:maxLength	限制显示的文本长度，超出部分不显示
android:lines	设置文本的行数，设置两行就显示两行，即使第二行没有数据
android:maxLines	设置文本的最大显示行数，与 width 或者 layout_width 结合使用，超出部分自动换行，超出行数将不显示
android:minLines	设置文本的最小行数，与 lines 类似
android:lineSpacingExtra	设置多行文本的行间距
android:lineSpacingMultiplier	设置行间距的倍数，例如"1.5"
android:numeric	如果被设置，限制 TextView 只能输入数字，例如 android:numeric="integer" 设置只能输入整数，如果是小数则是：decimal
android:password	设置文本框为密码输入框，以小点"."的外观显示文本，其取值为“true”或“false”，例如 android:password="true"，则设置只能输入密码
android:phoneNumber	设置为电话号码的输入方式，其取值为“true”或“false”
android:privateImeOptions	设置输入法选项
android:scrollHorizontally	设置文本超出 TextView 的宽度的情况下，是否出现横向滚动条
android:selectAllOnFocus	如果文本是可选择的，设置当它获取焦点时，是否自动选中所有文本

续表

属　　性	描　　述	
android:shadowColor	指定文本阴影的颜色，需要与 shadowRadius 一起使用	
android:shadowDx	设置阴影横向坐标开始位置	
android:shadowDy	设置阴影纵向坐标开始位置	
android:shadowRadius	设置阴影的半径。设置为 0.1 就变成字体的颜色了，一般设置为 3.0 的效果比较好	
android:singleLine	设置是否为单行显示，其取值为“true”或“false”。如果和 layout_width 一起使用，当文本不能全部显示时，后面用"…"来表示，例如对于 android:text="test_ singleLine "，如设置 android:singleLine="true"，android:layout_width="20dp"，则文字不会自动换行，将只显示"…"。如果不设置 singleLine 或者设置为 false，文本将自动换行	
android:text	设置控件要显示的文字。可以直接指定其为某个字符串，例如 android:text="姓名"；也可以让它引 res\value\strings.xml 字符串资源中的某个字符串，例如 android:text="@string/strname"	
android:textAllCaps	设置文本是否为大写，可选值为“true”或“false”	
android:textAlign	设置控件内容对齐	
android:textAppearanc	设置文字外观	
android:textColor	设置文本颜色，可以通过指定红、绿、蓝三种颜色值设定颜色值，例如 android:textColor = "#ff8c00"，三种颜色用十六进制方式指定，每种颜色的范围为 00~FF	
android:textColorHighlight	设置被选中文字的底色，默认为蓝色。 例如 android:textColorHighlight="#cccccc"	
android:textColorHint	设置提示信息文字的颜色，默认为灰色，与 hint 一起使用，例如 android:textColorHint="#ffff00"	
android:textColorLink	设置文字链接的颜色	
android:textIsSelectable	设置文本内容是否可编辑，其取值为"true"或"false"	
android:textScaleX	设置文字之间的间距，默认为 1.0f，例如 android:textScaleX="1.5"	
android:textSize	设置文字的字体大小，推荐度量单位为"sp"，例如 android:textSize="20dip"	
android:textStyle	设置文字的字体，可选值为 bold(粗体)-0、italic(斜体)-1、bolditalic(粗全+斜体)-2，可以设置一个或多个，用"	"隔开，例如 android:textStyle="bold"
android:typeface	设置文本字体，可选值为 normal – 0、sans – 1、serif – 2、monospace - 3 (等宽字体)，例如 android:typeface="monospace"	
android:height	设置控件的宽度，支持度量单位有 px(像素)、dp、sp、in、mm(毫米)	
android:width	设置控件的高度，支持度量单位有 px(像素)、dp、sp、in、mm(毫米)	
android:maxHeight	设置文本区域的最大高度	
android:maxWidth	设置文本区域的最大宽度	
android:minHeight	设置文本区域的最小高度	
android:minWidth	设置文本区域的最小宽度	

EditText 的基本属性如表 3-6 所示。

表 3-6　EditText 的基本属性

属　　性	描　　述
android:text	设置控件显示的文本
android:contentDescription	定义文本简要描述内容

续表

属　　性	描　　述
android:id	设置一个标识符名称
android:hint	设置 EditText 为空时显示的提示信息
android:gravity	设置控件文本内容的显示位置
android:layout_gravity	设置控件相对于父容器的显示位置，默认 top
android:onClick	控件被单击时调用的方法名称
android:visibility	控制控件的初始为可视性

3．重要方法

通过调用控件的方法可以动态修改 TextView 控件与 EditText 控件的属性。

① public CharSequence getText()

方法 getText()用于获取文本控件的文本内容，返回值为控件当前显示的字符串，通过方法 toString()将其转化为熟悉的 String 类型。

示例代码如下：

```
TextView tv=(TextView)findViewById(R.id.text);
String strText=tv.getText().toString();
```

② public final void setText(int resid)

该方法用于设置 TextView 的显示内容为某个字符串资源。

示例代码如下：

```
TextView tv=(TextView)findViewById(R.id.text);
tv.setText(R.string.username);
```

第 1 行代码使用 findViewById 方法通过控件的 id 获得控件的对象，R.id.text 是 TextView 控件的 id，tv 是该 TextView 控件对象，R.string.username 是某个字符串资源的 id。

③ public final void setText(CharSequence text)

该方法用于设置 TextView 的显示内容为参数给定的字符串。

示例代码如下：

```
TextView tv=(TextView)findViewById(R.id.text);
tv.setText("用户名");
```

4．示例代码

```
<TextView
    android:id="@+id/tvname"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="用户名"
    android:textSize="10pt" />
<EditText
    android:id="@+id/editname"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="请输入用户名"
    android:selectAllOnFocus="true" />
```

3.9 Button 控件

1．功能说明

Button 控件称按钮控件，Button 在界面上生成一个按钮，按钮可以供用户进行单击操作，单击按钮后一般会触发一系列处理。Button 按钮上通常显示文字，也可以通过 android:background 属性为 Button 增加背景颜色或背景图片。Button 类继承自 android.widget.TextView 类，TextView 控件的大部分属性和方法 Button 都可使用。在 android.widget 包中，其常用子类有 CheckBox、RadioButton 和 ToggleButton。

2．重要方法

Button 控件的重要方法如下：

① setClickable(boolean clickable)

设置按钮是否允许单击，取值为 true 时表示允许单击，取值为 false 时则禁止单击。

② setBackgroundResource(int resid)

通过资源文件设置背景色，resid 为资源 XML 文件的 id。按钮默认背景为 android.R.drawable.btn_default。

③ setText(CharSequence text)

设置按钮显示的文字内容。

④ setTextColor(int color)

设置按钮显示文字的颜色，color 可以使用系统 Color 常量，例如 Color.BLACK 等。

⑤ setOnClickListener(OnClickListener l)

设置按钮的单击事件。

3．示例代码

```
<Button
    android:layout_width="fill_parent"          //充满父控件
    android:layout_height="wrap_content"        //控件充满内容
    android:id="@+id/btnLogin"                  //设置 button 的 ID
    android:text="登录"  />                     //设置按钮的文本显示信息
<Button
    android:layout_width="150dip"               //按钮 2 的宽度
    android:layout_height="30dip"               //按钮 2 的高度
    android:background="#aa00aa"                //设置按钮 2 背景颜色
    android:textColor="#00aa00"                 //设置按钮 2 里的文本颜色
    android:layout_gravity="center"             //设置控件居中显示，
    //注意：android:gravity="center"表是文本在控件中居中显示
     android:id="@+id/btnRegister"
     android:text="注册"  />
```

3.10 DatePicker 控件与 TimePicker 控件

1．DatePicker 控件

DatePicker 控件用于选择日期，通过选择年、月、日，可以确定一个日期。DatePicker 继承自 FrameLayout 类，日期选择控件的主要功能是向用户提供包含年、月、日的日期数据并允许用户对其修改。如果要捕获用户修改日期选择控件中的数据事件，需要为 DatePicker 添

加 OnDateChangedListener 监听器。

DatePicker 控件的基本属性如下：

① android:startYear

设置 DatePicker 控件的起始年份，实际上就是该控件可以选择的最小年份。

② android:endYear

设置 DatePicker 控件的结束年份，实际上就是该控件可以选择的最大年份。

③ android:calendarViewShown

设置是否显示日历，取值为“true”时，显示日历，取值为“false”时，不显示日历。

DatePicker 控件的重要方法如下：

① public int getYear()

用于获取当前 DatePicker 控件选择的年份，返回当前选中的年份。

② public int getMonth()

用于获取当前 DatePicker 控件选择的月份，返回当前选中的月份。

③ public int getDayOfMonth()

用于获取当前 DatePicker 控件选择的日期，返回当前选中的日期。

④ public void updateDate(int year , int month , int dayOfMonth)

用于设置 DatePicker 控件所显示的年月日，其中 year 为年份，month 为月份（取值为 0～11，代表 1～12 月），dayOfMonth 为日。

2．TimePicker 控件

TimePicker 控件用于选择时间，常与 DatePicker 控件配合使用。TimePicker 继承自 FrameLayout 类。时间选择控件向用户显示一天中的时间（可以为 24 小时，也可以为 AM/PM 制），并允许用户进行选择。如果要捕获用户修改时间数据的事件，便需要为 TimePicker 添加 OnTimeChangedListener 监听器。

TimePicker 控件的重要方法如下：

① public Integer getCurrentHour)

用于获取当前控件选择的小时，返回当前选中的小时。

② public Integer getCurrentMinute()

用于获取当前控件选择的分钟，返回当前选中的分钟。

③ public void setCurrentHour(Integer currentHour)

用于设置当前控件显示的小时。

④ public void setCurrentMinute(Integer currentMinute)

用于设置当前控件显示的分钟。

⑤ public void setIs24HourView(Boolean is24HourView)

用于设置控件是否采用 24 小时制来显示时间。取值为 false 则使用 12 小时制，并有一个 am（上午）和 pm（下午）的选项；取值为 true 则使用 24 小时制，且没有 am 和 pm 的选项。

3．DatePicker 控件与 TimePicker 控件的示例代码

DatePicker 控件与 TimePicker 控件的示例代码如下所示。

```
<DatePicker
    android:id="@+id/datePicker"
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"  />
    <TimePicker
        android:id="@+id/timePicker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"  />
```

在程序中动态修改日期和时间的代码如下所示。

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.datetime);
        DatePicker dp = (DatePicker) findViewById(R.id.datePicker);
        TimePicker tp = (TimePicker) findViewById(R.id.timePicker);
        dp.init(2015,                          //年
                11,                            //月起点月份为0，所以11代表12月
                28,                            //日
                null);
        tp.setIs24HourView(true);              //使用24小时格式
        tp.setCurrentHour(new Integer(10));    //设置当前小时
         tp.setCurrentMinute(new Integer(18)); //设置分钟
        }
```

上面代码设置的日期为 2015-12-28，时间为 10:18。如果我们没有在代码中修改时间，默认显示的是手机当前的日期和时间。Android 还提供了 DatePickerDialog 和 TimePickerDialog 两个对话框让用户可以在对话框中挑选日期和时间，避免这两个控件占用太多的空间。

3.11 Android 资源应用

编程时，大部分时候提到的 Android 资源应用，都是指位于 res 目录下的应用资源，Android SDK 会在编译该应用时在 R 类中为它们创建对应的索引项。

1．资源的类型及存储方式

Android 要求在 res 目录下使用不同的子文件夹保存不同的应用资源，表 3-7 所示为 Android 不同资源在/res 目录下的存储方式。

表 3-7　Android 不同资源在/res 目录下的存储方式

文　件　夹	存放的资源
/res/anim/	存放定义补间动画的 XML 文件
/res/color/	存放定义不同状态下颜色列表的 XML 文件
/res/drawable/	该文件下存放各种位图文件（如*.png、*.jpg*、.gif）等，除此之外还可以编译成如下各种 Drawable 对象的 XML 文件：BitmapDrawable 对象、NinePatchDrawable 对象、StateListDrawable 对象、ShapeDrawable 对象、AnimationDrawable 对象、Drawable 的其他各种子类的对象
/res/layout/	存放各种用户界面的布局文件
/res/menu/	存放为应用程序定义各种菜单的资源，包括选项菜单、子菜单、上下文菜单资源
/res/raw/	该文件夹下存放任意类型的原生资源。在 Java 代码中通过调用 Resource 对象的 openRawResource（int id）方法获取该资源的二进制输入流。实际上，如果应用程序使用原生资源，推荐把这些原生资源保存到/assrts 目录下，然后在应用程序中使用 AssetManager 来访问这些资源

续表

文　件　夹	存放的资源
/res/values/	存放各种简单的 XML 文件。这些简单值包括字符串值、整数值、颜色值、数组等。 字符串、整数值、颜色值、数组等各种值都存放在该文件夹下，而且这些资源文件的根文件夹都是<resources…/>元素，为该<resource…./>元素添加不同的子元素则代表不同的资源，例如：string/integer/bool 子元素：代表添加一个字符串值/整数值/boolean 值；color 子元素：代表添加一个颜色值；array 子元素或 string-array、int-array 子元素：代表添加一个数组；style 子元素：代表添加一个样式；dimen：代表添加一个尺寸 由于各种简单值都可以定义在/res/values/目录下的资源文件中，如果在同一份资源文件中定义各种值，势必增加程序维护的难度。为此，Android 建议使用不同的文件来存放不同类型的值：① arrays.xml：定义数组资源；② colors.xml：定义颜色值资源；③ dimens.xml：定义尺寸值资源；④ strings.xml：定义字符串资源；⑤ styles.xml：定义样式资源
/res/xml/	存放任意的原生 XML 文件，这些 XML 文件可在 Java 代码中使用 Resources.getXML()访问

2．使用资源

（1）在 Java 代码中使用资源

由于 Android SDK 会在编译应用时在 R 类中为/res 文件夹下所有资源创建索引项，因此在 Java 代码中访问资源主要通过 R 类来完成。其完整的语法格式为：

```
[<package_name>.]R.<resource_type>.<resource_name>;
```

各个参数的含义如下：

① <package_name>：指定 R 类所在包，实际上就是使用全限定类名。当然，如果在 Java 程序中导入 R 类所在包，就可以省略包名。

② <resource_type>：R 类中代表不同资源类型的子类，例如 string 代表字符串资源。

③ <resource_name>：指定资源的名称。该资源名称可能是无后缀的文件名（如图片资源），也可能是 XML 资源文件中由 android：name 属性所指定的名称。

（2）在 XML 代码中使用资源

当定义 XML 资源文件时，其中的 XML 元素可能需要指定不同的值，这些值就可以设置为已定义的资源项，在 XML 中使用资源的完整语法格式为：

```
@[<package_name>:]<resource_type>/<resource_name>;
```

各个参数的含义如下：

① <package_name>：指定资源类所在应用包，如果引用的资源和当前资源位于同一个包下，则可以省略。

② <resource_type>：R 类中代表不同资源类型的子类，例如 string 代表字符串资源。

③ <resource_name>：指定资源的名称。该资源名称可能是无后缀的文件名（如图片资源），也可能是 XML 资源文件中由 android：name 属性所指定的名称。

3．使用字符串、颜色、尺寸资源

字符串资源、颜色资源、尺寸资源，它们对应的 XML 文件都位于/res/values 目录下，它们默认的文件名，以及在 R 类中对应的内部类如表 3-8 所示。

表 3-8　字符串资源、颜色资源、尺寸资源说明

资源类型	资源文件的默认名	对应于 R 类中的内部类的名称
字符串资源	/res/values/strings.xml	R.string
颜色资源	/res/values/colors.xml	R.color
尺寸资源	/res/values/dimens.xml	R.dimen

Android 允许使用资源文件定义 boolean 常量，默认名为：/res/values/bools.xml，该文件的根元素也是<resources…/>，根元素内通过<bool…/>子元素定义 boolean 常量，对应 R 类中内部类的名称为 R.bool。

Android 允许使用资源文件定义整型常量，默认名为：/res/values/integers.xml，该文件的根元素也是<resources…/>，根元素内通过<integer…/>子元素定义整型常量，对应 R 类中内部类的名称为 R.integer。

4．数组（Array）资源

Android 采用位于/res/values 目录下的 arrays.xml 文件来定义数组，定义数组时 XML 资源文件的根目录也是<resources…/>元素，该元素内可包含如下三种子元素：

① <array…/>子元素：定义普通类型的数组，例如 Drawable 数组。

② <string-array…/>子元素：定义字符串数组。

③ <integer-array…/>子元素：定义整数数组。

在 Java 代码中的访问形式：[<package_name>.]R.array.array_name

在 XML 代码中的访问形式：@[<package_name>:]array/array_name

为了能在 Java 程序中访问到实际数组，Resources 提供了如下方法：

① String[] getStringArray（int id）：根据资源文件中字符串数组资源的名称来获取实际的字符串数组。

② int[] getIntArray（int id）：根据资源文件中整型数组资源的名称来获取实际的整型数组。

③ TypedArray obtainTypedArray（int id）：根据资源文件中普通数组资源的名称来获取实际的普通数组。TypedArray 代表一个通用类型的数组，该类提供了 getXxx（int index）方法来获取指定索引处的数组元素。



【任务 3-1】使用文字标签显示欢迎信息



【任务描述】

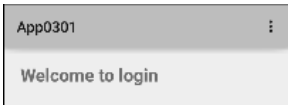


图 3-9　Android 项目 App0301 的运行结果

创建 Android 项目 App0301，在 strings.xml 文件中定义字符串“loginInfo”，设置 TextView 控件的 text、宽度、高度、padding、文字大小、文字加粗、文字居中等多项属性，控件属性设置完成浏览布局文件 activity_main.xml 和 MainActivity.java 的代码，然后运行程序，运行结果如图 3-9 所示。

【知识索引】

（1）在 strings.xml 文件中定义字符串。

（2）RelativeLayout 布局。

（3）TextView 控件及其属性设置。

【实施过程】

1．创建 Android 项目 App0301

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0301。

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加新的字符串定义，字符串名称为“loginInfo”，其值为“Welcome to login”，strings.xml 文件的代码如表 3-9 所示。

表 3-9　strings.xml 文件的代码

序号	字符串定义代码
01	<resources>
02	<string name="app_name">App0301</string>
03	<string name="hello_world">Hello world!</string>
04	<string name="action_settings">Settings</string>
05	<string name="loginInfo">Welcome to login</string>
06	</resources>

3．添加控件与设置其属性

（1）查看布局文件 activity_main.xml 中自动生成的代码

打开布局文件 activity_main.xml，系统自动添加的代码如表 3-10 所示，可以看出界面的布局方式为相对布局（RelativeLayout），并且自动添加了 1 个 TextView 控件，该控件的 Text 属性值为“Hello world!”。

表 3-10　布局文件 activity_main.xml 自动添加的代码

序号	布 局 代 码
01	<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	xmlns:tools="http://schemas.android.com/tools"
03	android:layout_width="match_parent"
04	android:layout_height="match_parent"
05	android:paddingLeft="@dimen/activity_horizontal_margin"
06	android:paddingRight="@dimen/activity_horizontal_margin"
07	android:paddingTop="@dimen/activity_vertical_margin"
08	android:paddingBottom="@dimen/activity_vertical_margin"
09	tools:context=".MainActivity">
10	<TextView android:text="@string/hello_world"
11	android:layout_width="wrap_content"
12	android:layout_height="wrap_content" />
13	</RelativeLayout>

（2）删除默认添加的 TextView 控件

为了说明添加控件的过程，这里先将默认添加的 TextView 控件删除，在编辑工作区单击【Design】标签，切换到设计视图，在设计界面选中已有的 TextView 控件，被选中的控件周围会出现 8 个小正方形，如图 3-10 所示。然后按【Delete】键删除该控件。



图 3-10　设计界面选中控件

（3）重新添加 1 个 TextView 控件

Android Studio 提供了非常丰富的各类控件，包括 Widgets（如图 3-11 所示）、Text Fields（如图 3-12 所示）、Containers（如图 3-13 所示）、Date & Time（如图 3-14 所示）、Expert（如图 3-15 所示）、Custom（如图 3-16 所示）六种类型，这六种类型的控件又包含了多种控件，为了快速设置程序界面，对这些控件的外观和功能应熟悉。

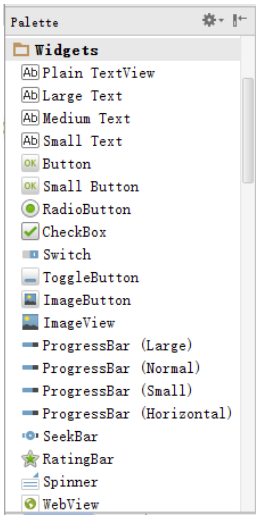


图 3-11　Widgets 类控件

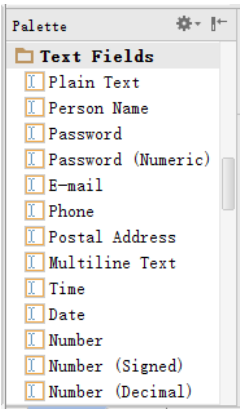


图 3-12　Text Fields 类控件

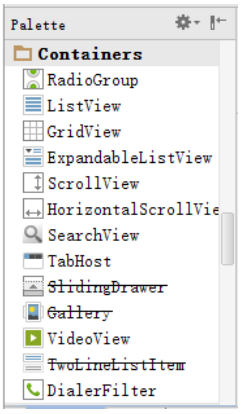


图 3-13　Containers 类控件

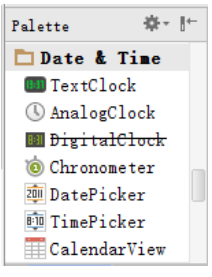


图 3-14　Date & Time 类控件

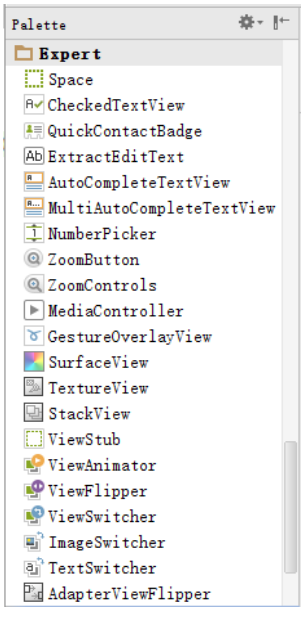


图 3-15　Expert 类控件

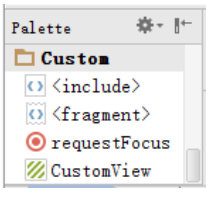


图 3-16　Custom 类控件

这里需要添加 1 个 TextView 控件，在【Palette】面板的“Widgets”区域单击控件“Plain TextView”，拖动鼠标进入界面设计区域，将显示一个控件虚线框、相对于左边框和上边框的定位箭头以及距离数值，如图 3-17 所示，图中 alignParentLeft 表示相对于父布局左对齐，alignParentTop 表示相对于父布局顶部对齐。

一边拖动鼠标，一边观察控件位置，位置合适时单击鼠标左键，TextView 控件便添加完成。

（4）设置 TextView 控件的属性

在设计界面选中已添加的 TextView 控件，然后在“Properties”区域找到“text”属性，单击其右侧的【…】按钮，打开“Resources”对话框，在该对话框中选择 strings.xml 文件中已定义字符串“loginInfo”，如图 3-18 所示，然后单击【OK】按钮，完成 text 属性的设置。

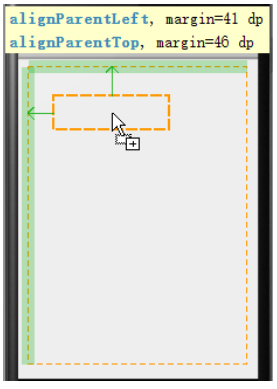


图 3-17　在界面设计区域添加 1 个控件

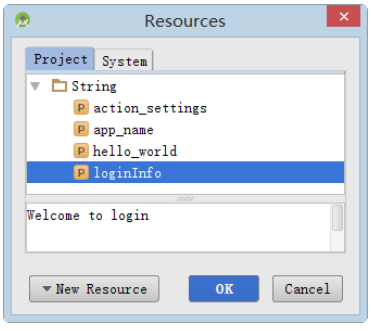


图 3-18　“Resources”对话框

接着在“Properties”区域设置 TextView 控件的宽度、高度、padding、文字大小、文字加粗、文字居中等多项属性，属性设置完成后的“Component Tree”和“Properties”区域如图 3-19 所示。

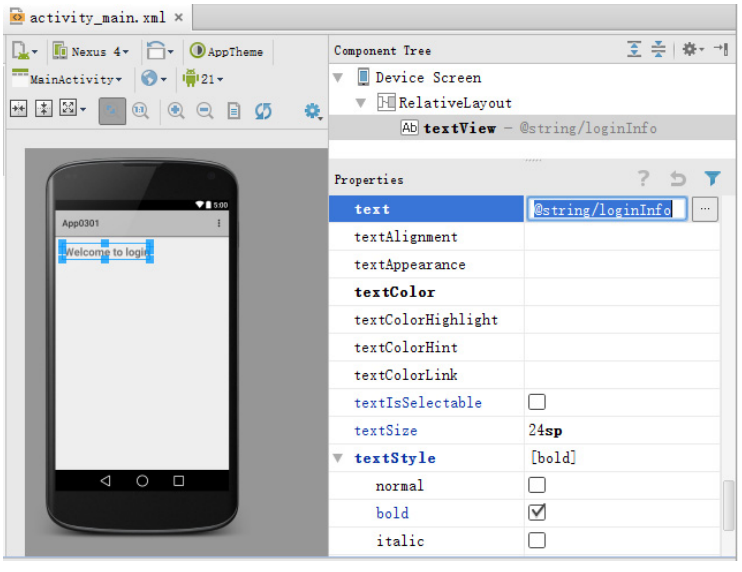


图 3-19　设置界面控件属性

4．查看布局文件 activity_main.xml 的代码

添加必要的控件，并完成控件属性设置后，布局文件 activity_main.xml 的代码如表 3-11 所示。

表 3-11　布局文件 activity_main.xml 的代码

序号	布 局 代 码
01	<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	xmlns:tools="http://schemas.android.com/tools"
03	android:layout_width="match_parent"
04	android:layout_height="match_parent"
05	android:paddingLeft="@dimen/activity_horizontal_margin"
06	android:paddingRight="@dimen/activity_horizontal_margin"
07	android:paddingTop="@dimen/activity_vertical_margin"
08	android:paddingBottom="@dimen/activity_vertical_margin"
09	tools:context=".MainActivity">
10	<TextView android:text="@string/loginInfo"
11	android:layout_width="wrap_content"
12	android:layout_height="wrap_content"
13	android:id="@+id/textView"
14	android:textSize="24sp"
15	android:textStyle="bold"
16	android:gravity="center"
17	android:visibility="visible"
18	android:textIsSelectable="false"
19	android:paddingLeft="5dp"
20	android:paddingTop="5dp"
21	android:paddingRight="5dp"
22	android:paddingBottom="5dp" />
23	</RelativeLayout>

布局文件 activity_main.xml 中包含了多个样式设置，Android 常用的样式设置说明如下：

（1）设置元素宽度：android:layout_width。

（2）设置元素高度：android:layout_height。

（3）设置元素 padding，包括 android:paddingLeft、android:paddingRight、android:paddingTop、android:paddingBottom。

例如：

```
android:paddingTop="5dp"
android:paddingBottom="5dp"
android:paddingLeft="5dp"
android:paddingRight="5dp"
```

（4）设置文字大小：android:textSize，例如：android:textSize="24sp"。

（5）设置文字加粗：android:textStyle，例如：android:textStyle="bold"。

（6）设置文字居中：android:gravity，例如：android:gravity="center"。

（7）设置元素不显示：android:visibility，例如：android:visibility="invisible"。

（8）设置文字可选择：android:textIsSelectable，例如：android:textIsSelectable="false"。

dimens.xml 文件的代码如表 3-12 所示，表中定义的字符串 activity_horizontal_margin 和 activity_vertical_margin 的值为“16dp”。

表 3-12　dimens.xml 文件中的代码

序号	代　码
01	<resources>
02	<!-- Default screen margins, per the Android Design guidelines. -->
03	<dimen name="activity_horizontal_margin">16dp</dimen>
04	<dimen name="activity_vertical_margin">16dp</dimen>
05	</resources>

5．程序运行与查看结果

Android 项目 App0301 的运行结果如图 3-9 所示。

【任务 3-2】 设计包含多种控件的用户登录界面

【任务描述】

创建 Android 项目 App0302，设计包含多种控件的用户登录界面，在 strings.xml 文件中定义多个字符串，在布局文件中添加多个控件，并设置各个控件的属性，该项目在“Preview”面板中的预览效果如图 3-20 所示。



图 3-20　Android 项目 App0302 的预览效果

【知识索引】

（1）在 strings.xml 文件中定义字符串。

（2）RelativeLayout 布局。

（3）TextView、EditText、CheckBox、RadioGroup、RadioButton、Button 等控件以及各个控件的属性设置。

【实施过程】

1．创建 Android 项目 App0302

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0302。

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加多个字符串定义，strings.xml 文件的代码如表 3-13 所示。

表 3-13　字符串定义文件 strings.xml 的代码

序号	布 局 代 码
01	<resources>
02	<string name="app_name">App0302</string>
03	<string name="hello_world">Hello world!</string>

续表

序号	布 局 代 码
04	<string name="action_settings">Settings</string>
05	<string name="loginInfo">Welcome to login</string>
06	<string name="InfoPasswordInput">请输入密码</string>
07	<string name="LoginName">Name</string>
08	<string name="InfoNameInput">请输入用户名</string>
09	<string name="chkLogin">下次自动登录</string>
10	<string name="rbtnLogin1">二维码登录</string>
11	<string name="rbtnLogin2">邮箱账号登录</string>
12	<string name="btnLogin">登录</string>
13	<string name="btnExit">退出</string>
14	</resources>

3．完善布局文件 activity_main.xml 与界面设计

项目创建时默认添加了 1 个 TextView 控件，在“Properties”区域设置“text”属性值为已定义字符串“loginInfo”。

然后添加 1 个 EditText 控件，在【Palette】面板的“Text Fields”区域单击控件“Plain Text”，拖动鼠标进入界面设计区域，将显示一个控件虚线框、相对定位箭头以及对齐方式，如图 3-21 所示，这里相对定位可以相对于设计区域的边框定位，也可以相对于已添加控件的定位，图中控件相对于左边框左对齐，距离为 69dp，相对已添加控件顶部对齐，距离为 28dp。

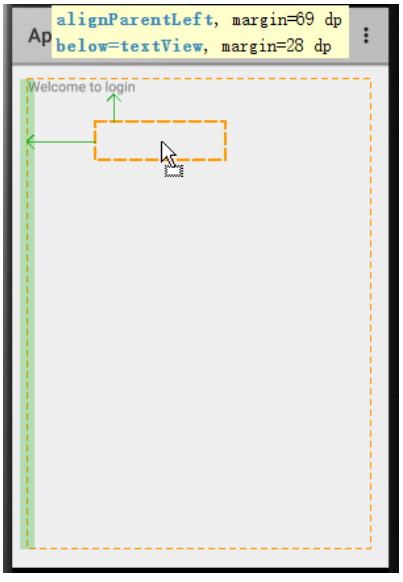


图 3-21　在界面中添加第 2 个控件

接着按类似方法添加 1 个 EditText 控件、1 个 CheckBox 控件、1 个 RadioGroup 控件和 2 个 Button 控件，在 RadioGroup 控件控件中添加 2 个 RadioButton 控件，并设置好各个控件的属性。布局文件 activity_main.xml 中控件对应的代码如表 3-14 所示。

表 3-14　布局文件 activity_main.xml 中控件对应的代码

序号	布 局 代 码
01	<RelativeLayout>
02	<TextView android:text="@string/loginInfo"
03	android:layout_width="wrap_content"
04	android:layout_height="wrap_content"
05	android:layout_alignParentTop="true"
06	android:id="@+id/textView" />
07	<EditText
08	android:layout_width="wrap_content"
09	android:layout_height="wrap_content"
10	android:inputType="textPersonName"
11	android:text="@string/LoginName"
12	android:ems="10"
13	android:id="@+id/editText1"
14	android:layout_below="@+id/textView"
15	android:layout_alignParentStart="true"
16	android:layout_marginStart="44dp"
17	android:layout_marginTop="31dp"
18	android:hint="@string/InfoNameInput"
19	android:typeface="normal" />
20	<EditText
21	android:layout_width="wrap_content"
22	android:layout_height="wrap_content"
23	android:inputType="textPassword"
24	android:ems="10"
25	android:id="@+id/editText2"
26	android:layout_below="@+id/editText1"
27	android:layout_alignStart="@+id/editText1"
28	android:layout_marginTop="35dp"
29	android:selectAllOnFocus="false"
30	android:hint="@string/InfoPasswordInput" />
31	<CheckBox
32	android:layout_width="wrap_content"
33	android:layout_height="wrap_content"
34	android:text="@string/chkLogin"
35	android:id="@+id/checkBox"
36	android:layout_below="@+id/editText2"
37	android:layout_alignStart="@+id/editText2"
38	android:layout_marginTop="30dp" />
39	<RadioGroup
40	android:layout_width="wrap_content"
41	android:layout_height="wrap_content"

续表

序号	布 局 代 码
42	android:id="@+id/rgLogin"
43	android:orientation="horizontal"
44	android:layout_below="@+id/checkBox"
45	android:layout_alignEnd="@+id/editText2"
46	android:layout_marginTop="31dp">
47	<RadioButton
48	android:layout_width="wrap_content"
49	android:layout_height="wrap_content"
50	android:text="@string/rbtnLogin1"
51	android:id="@+id/radioButton1" />
52	<RadioButton
53	android:layout_width="wrap_content"
54	android:layout_height="wrap_content"
55	android:text="@string/rbtnLogin2"
56	android:id="@+id/radioButton2"
57	android:layout_gravity="center_horizontal" />
58	</RadioGroup>
59	<Button
60	android:layout_width="wrap_content"
61	android:layout_height="wrap_content"
62	android:text="@string/btnLogin"
63	android:id="@+id/button1"
64	android:layout_below="@+id/rgLogin"
65	android:layout_alignStart="@+id/rgLogin"
66	android:layout_marginTop="38dp" />
67	<Button
68	android:layout_width="wrap_content"
69	android:layout_height="wrap_content"
70	android:text="@string/btnExit"
71	android:id="@+id/button2"
72	android:layout_alignBottom="@+id/button1"
73	android:layout_toEndOf="@+id/checkBox" />
74	</RelativeLayout>

4．预览用户登录界面的设计效果

Android 项目 App0302 在 Preview 面板中的预览效果如图 3-20 所示。

【任务 3-3】 设计开关与调节声音的界面

【任务描述】

创建 Android 项目 App0303，设计开关与调节声音的界面，在 strings.xml 文件中定义多

个字符串，在布局文件中添加多个控件，并设置各个控件的属性，该项目的初始运行结果如图 3-22 所示。



图 3-22　Android 项目 App0303 的初始运行结果

【知识索引】

（1）在 strings.xml 文件中定义字符串。

（2）RelativeLayout 布局。

（3）TextView、ToggleButton、SeekBar 等控件及其属性设置。

【实施过程】

1．创建 Android 项目 App0303

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0303。

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加多个字符串定义，strings.xml 文件的代码如表 3-15 所示。

表 3-15　字符串定义文件 strings.xml 的代码

序号	布 局 代 码
01	<resources>
02	<string name="app_name">App0303</string>
03	<string name="action_settings">Settings</string>
04	<string name="tbTextOn">声音开启</string>
05	<string name="tbTextOff">声音关闭</string>
06	<string name="tvInfo1">音量调节</string>
07	<string name="tbtn">ToggleButon</string>
08	</resources>

3．完善布局文件 activity_main.xml 与界面设计

完善项目 App0303 的 res/layout 文件夹下的布局文件 activity_main.xml，除了项目创建时默认添加的 TextView 控件外，另外添加 1 个 ToggleButton 控件和 1 个 SeekBar 控件，并设置好各个控件的属性。布局文件 activity_main.xml 中控件对应的代码如表 3-16 所示。

表 3-16　布局文件 activity_main.xml 中控件对应的代码

序号	布 局 代 码
01	<RelativeLayout …… >
02	<TextView
03	……

续表

序号	布 局 代 码
04	android:textAppearance="?android:attr/textAppearanceMedium"
05	android:text="@string/tvInfo1"
06	android:id="@+id/textView"
07	android:layout_alignParentTop="true"
08	android:layout_alignStart="@+id/seekBar" />
09	<ToggleButton
10	……
11	android:text="@string/tbtn"
12	android:id="@+id/toggleButton"
13	android:textOff="@string/tbTextOff"
14	android:textOn="@string/tbTextOn"
15	android:layout_below="@+id/seekBar"
16	android:layout_alignStart="@+id/seekBar" />
17	<SeekBar
18	……
19	android:id="@+id/seekBar"
20	android:progress="10"
21	android:layout_alignParentStart="true"
22	android:layout_marginStart="47dp"
23	android:mirrorForRtl="true"
24	android:longClickable="true"
25	android:layout_below="@+id/textView"
26	android:layout_alignParentEnd="true" />
27	</RelativeLayout>

4．程序运行与查看结果

Android 项目 App0303 的初始运行结果如图 3-22 所示，拖动音量调节滑块，然后单击【声音关闭】按钮后结果如图 3-23 所示。



图 3-23　在界面中拖动音量调节滑块与单击【声音关闭】的结果

【任务 3-4】 使用 DatePicker 控件获取日期

【任务描述】

创建 Android 项目 App0304，使用 DatePicker 控件获取日期，该项目的运行结果如图 3-24 所示。



图 3-24　Android 项目 App0304 的运行结果

【知识索引】

（1）RelativeLayout 布局。

（2）DatePicker 控件及其属性设置。

【实施过程】

1．创建 Android 项目 App0304

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0304。

2．完善布局文件 activity_main.xml 与界面设计

先将默认添加的 TextView 控件删除，然后添加 1 个 DatePicker 控件，将 DatePicker 控件的 id 属性设置为 datePicker、layout_marginTop 属性设置为"10dp"、calendarViewShown 属性设置为"true"、spinnersShown 属性设置为"true"。

3．程序运行与查看结果

Android 项目 App0304 的运行结果如图 3-24 所示。

【任务 3-5】使用日期时间控件获取与显示时间

【任务描述】

创建 Android 项目 App0305，使用日期时间控件获取与显示时间，该项目的运行结果如图 3-25 所示。

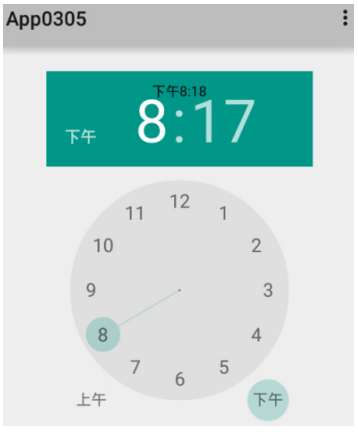


图 3-25　Android 项目 App0305 的运行结果

【知识索引】

（1）RelativeLayout 布局。

（2）TimePicker 控件、TextClock 控件及其属性设置。

TextClock 控件用于时间的文本显示，TextClock 不仅能显示时间，还能显示日期；而且支持自定义格式。

【实施过程】

1．创建 Android 项目 App0305

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0305。

2．完善布局文件 activity_main.xml 与界面设计

先将默认添加的 TextView 控件删除，然后添加 1 个 TimePicker 控件和 1 个 TextClock，并设置好该控件的属性。布局文件 activity_main.xml 中控件对应的代码如表 3-17 所示。

表 3-17　布局文件 activity_main.xml 中控件对应的代码

序号	布 局 代 码
01	<RelativeLayout>
02	<TimePicker
03	android:layout_width="wrap_content"
04	android:layout_height="wrap_content"
05	android:id="@+id/timePicker"
06	android:layout_alignParentTop="true"
07	android:layout_centerHorizontal="true"
08	android:layout_marginTop="10dp" />
09	<TextClock
10	android:layout_width="wrap_content"
11	android:layout_height="wrap_content"
12	android:id="@+id/textClock"
13	android:layout_alignParentTop="true"
14	android:layout_centerHorizontal="true"
15	android:layout_marginTop="20dp" />
16	</RelativeLayout>

3．程序运行与查看结果

Android 项目 App0305 的运行结果如图 3-25 所示。

【任务 3-6】使用 LinearLayout 布局设计用户注册界面

【任务描述】

创建 Android 项目 App0306，使用 LinearLayout 布局设计用户注册界面，在 strings.xml 文件中定义多个字符串，在布局文件中添加多个控件，并设置各个控件的属性，该项目在 Preview 面板中的预览效果如图 3-26 所示。

【知识索引】

（1）在 strings.xml 文件中定义字符串。

（2）LinearLayout 布局。

（3）TextView 控件、EditText 控件、Button 控件及各控件的属性设置。

【实施过程】

1．创建 Android 项目 App0306

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0306。



图 3-26　Android 项目 App0306 的预览效果

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加多个字符串定义，strings.xml 文件的代码如表 3-18 所示。

表 3-18　字符串定义文件 strings.xml 的代码

序号	布 局 代 码
01	<resources>
02	<string name="app_name">App0306</string>
03	<string name="action_settings">Settings</string>
04	<string name="LoginName">LoginName</string>
05	<string name="InfoLoginName">请输入用户名</string>
06	<string name="InfoPassword">请输入密码</string>
07	<string name="InfoPassword2">请再次输入密码</string>
08	<string name="TextLogin">登录</string>
09	</resources>

3．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 3 个 EditText 控件和 1 个 Button 控件，将 EditText 控件的 id 属性分别设置为 edtLoginName、edtPassword、edtPassword2，将 Button 控件的 id 属性设置为 btnLogin。

4．预览用户登录界面的设计效果

Android 项目 App0306 在 Preview 面板中的预览效果如图 3-26 所示。

【任务 3-7】 使用 TableLayout 布局设计商品信息展示界面

【任务描述】

创建 Android 项目 App0307，使用 TableLayout 布局设计商品信息展示界面，在 strings.xml

文件中定义多个字符串，在布局文件中添加多个 TableRow 对象和 TextView 控件，并设置其属性，该项目在 Preview 面板中的预览效果如图 3-27 所示。



图 3-27　Android 项目 App0307 的预览效果

【知识索引】

（1）在 strings.xml 文件中定义字符串。

（2）TableLayout 布局与 TableRow 对象。

（3）TextView 控件及其属性设置。

【实施过程】

1．创建 Android 项目 App0307

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0307。

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加多个字符串定义，strings.xml 文件的代码如表 3-19 所示。

表 3-19　字符串定义文件 strings.xml 的代码

序号	布 局 代 码
01	<resources>
02	<string name="app_name">App0307</string>
03	<string name="action_settings">Settings</string>
04	<string name="Title">商品信息</string>
05	<string name="order">序号</string>
06	<string name="name">商品名称</string>
07	<string name="price">价格</string>
08	</resources>

3．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为表格布局方式（TableLayout），且将默认添加的 TextView 控件删除，然后添加 4 个 TableRow 对象，即布局中有 4 行，第 1 行中添加 1 个 TextView 控件，第 2 行至第 4 行各添加 3 个 TextView 控件，并设置好各个控件的属性。布局文件 activity_main.xml 的代码如表 3-20 所示。

表 3-20　布局文件 activity_main.xml 的代码

序号	布 局 代 码
01	<TableLayout …… >
02	<!-- 定义 TableRow 控件，代表第 1 行 -->

续表

序号	布 局 代 码
03	<TableRow android:orientation="horizontal" >
04	<TextView
05	……
06	android:text="@string/Title"
07	android:id="@+id/tvTitle"
08	android:gravity="center"
09	android:textStyle="bold"
10	android:textSize="20sp"
11	android:layout_column="1"
12	android:layout_weight="1"/>
13	</TableRow>
14	<!-- 定义 TableRow 控件，代表第 2 行 -->
15	<TableRow android:orientation="horizontal" >
16	<TextView
17	……
18	android:gravity="center"
19	android:layout_weight="1"
20	android:text="@string/order" />
21	<TextView
22	……
23	android:layout_weight="1"
24	android:gravity="center"
25	android:text="@string/name" />
26	<TextView
27	……
28	android:layout_weight="1"
29	android:gravity="center"
30	android:text="@string/price" />
31	</TableRow>
32	<!-- 定义 TableRow 控件，代表第 3 行 -->
33	<TableRow android:orientation="horizontal" >
34	<TextView
35	……
36	android:layout_weight="1"
37	android:gravity="center"
38	android:text="1" />
39	<TextView
40	……
41	android:layout_weight="1"
42	android:gravity="center"
43	android:text="苹果 iPhone 6" />
44	<TextView

续表

序号	布 局 代 码
45	android:layout_width="wrap_content"
46	android:layout_height="wrap_content"
47	android:layout_weight="1"
48	android:gravity="center"
49	android:text="￥6188" />
50	</TableRow>
51	<!-- 定义 TableRow 控件，代表第 4 行 -->
52	<TableRow android:orientation="horizontal" >
53	<TextView
54	android:layout_width="wrap_content"
55	android:layout_height="wrap_content"
56	android:layout_weight="1"
57	android:gravity="center"
58	android:text="2" />
59	<TextView
60	android:layout_width="wrap_content"
61	android:layout_height="wrap_content"
62	android:layout_weight="1"
63	android:gravity="center"
64	android:text="三星 Galaxy Note 3" />
65	<TextView
66	android:layout_width="wrap_content"
67	android:layout_height="wrap_content"
68	android:layout_weight="1"
69	android:gravity="center"
70	android:text="￥2899" />
71	</TableRow>
72	</TableLayout>

4．预览用户登录界面的设计效果

Android 项目 App0307 在 Preview 面板中的预览效果如图 3-27 所示。

【任务 3-8】 使用 GridLayout 布局设计仿 Windows 8 桌面

【任务描述】

创建 Android 项目 App0308，使用 GridLayout 布局设计仿 Windows 8 桌面，在布局文件中添加多个 TextView 控件，并设置各个控件的属性，该项目在 Preview 面板中的预览效果如图 3-28 所示。

【知识索引】

（1）GridLayout 布局。

（2）TextView 控件及其属性设置。



图 3-28　Android 项目 App0308 的预览效果

【实施过程】

1．创建 Android 项目 App0308

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0308。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为网格布局方式（GridLayout），且将默认添加的 TextView 控件删除，然后添加 5 个 TextView 控件，并设置好各个控件的属性。布局文件 activity_main.xml 的代码如表 3-21 所示。

表 3-21　布局文件 activity_main.xml 的代码

序号	布 局 代 码
01	<GridLayout ……
02	android:columnCount="4">
03	<TextView
04	android:layout_width="200dp"
05	android:layout_height="120dp"
06	android:layout_columnSpan="2"
07	android:layout_gravity="left"
08	android:background="#0E8257"
09	android:gravity="center"
10	android:text="旅游指南"
11	android:width="50dp" />
12	<TextView
13	android:layout_width="160dp"
14	android:layout_height="120dp"
15	android:layout_columnSpan="2"
16	android:background="#FAC040"
17	android:gravity="center"
18	android:text="火车票"
19	android:width="50dp"
20	android:layout_gravity="top" />
21	<TextView
22	android:layout_width="110dp"

续表

序号	布 局 代 码
23	android:layout_height="120dp"
24	android:layout_gravity="top"
25	android:background="#66C2F7"
26	android:gravity="center"
27	android:text="机票"
28	android:width="25dp" />
29	<TextView
30	android:layout_width="150dp"
31	android:layout_height="120dp"
32	android:layout_columnSpan="2"
33	android:background="#94C314"
34	android:gravity="center"
35	android:text="团购"
36	android:width="50dp" />
37	<TextView
38	android:layout_width="100dp"
39	android:layout_height="120dp"
40	android:background="#0D7BBA"
41	android:gravity="center"
42	android:text="酒店"
43	android:width="25dp" />
44	</GridLayout>

3．预览用户登录界面的设计效果

Android 项目 App0308 在 Preview 面板中的预览效果如图 3-28 所示。

【任务 3-9】 使用 FrameLayout 布局实现图片相框效果

【任务描述】

创建 Android 项目 App0309，使用 FrameLayout 布局实现图片相框效果，在布局文件中添加多个 ImageView 控件，并设置各个控件的属性，该项目在 Preview 面板中的预览效果如图 3-29 所示。

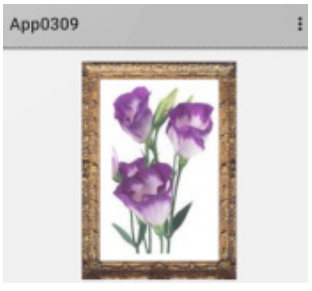


图 3-29　Android 项目 App0309 的预览效果

【知识索引】

（1）FrameLayout 布局。

（2）ImageView 控件及其属性设置。

【实施过程】

1．创建 Android 项目 App0309 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0309，将本任务所需的图片文件导入或复制到 res/drawable 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为帧布局方式（FrameLayout），且将默认添加的 TextView 控件删除，然后添加 2 个 ImageView 控件，并设置好各个控件的属性。布局文件 activity_main.xml 的代码如表 3-22 所示。

表 3-22　布局文件 activity_main.xml 的代码

序号	布 局 代 码
01	<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	xmlns:tools="http://schemas.android.com/tools"
03	android:layout_width="match_parent"
04	android:layout_height="match_parent"
05	android:paddingLeft="@dimen/activity_horizontal_margin"
06	android:paddingRight="@dimen/activity_horizontal_margin"
07	android:paddingTop="@dimen/activity_vertical_margin"
08	android:paddingBottom="@dimen/activity_vertical_margin"
09	tools:context=".MainActivity">
10	<ImageView
11	android:layout_width="wrap_content"
12	android:layout_height="wrap_content"
13	android:src="@drawable/imgbg"
14	android:layout_gravity="center_horizontal\|top" />
15	<ImageView
16	android:layout_width="162dp"
17	android:layout_height="239dp"
18	android:padding="20dp"
19	android:src="@drawable/flower"
20	android:layout_marginLeft="6dp"
21	android:layout_marginTop="10dp"
22	android:layout_gravity="center_horizontal\|top" />
23	</FrameLayout>

3．预览用户登录界面的设计效果

Android 项目 App0309 在 Preview 面板中的预览效果如图 3-29 所示。



用户界面是 Android 应用程序的重要组成部分，Android 系统中提供了许多控件，包括文本控件、按钮控件、图片控件、时间控件等，控件可以为应用程序构建美观、易用的用户界面。本单元主要介绍了 Android 屏幕元素的层次结构、View 与 ViewGroup、Android 的布局对象、Android 中创建 UI 界面的方式、Android 常用的 UI 控件及其基本属性、Android 尺寸的单位等内容，重点探析了 TextView、EditText、Button、DatePicker、TimePicke 等常用控件的功能、基本属性以及重要方法。通过完成多个界面的设计任务，学会灵活运用控件和布局对

象设置美观的用户界面。



单元习题

一、填空题

1. Android 中的布局分为6种，分别是（　　）、（　　）、（　　）、（　　）、FrameLayout 和 AbsoluteLayout。

2．Android 相对布局中，表示“是否跟父布局左对齐”的属性是（　　）。

3．相关性布局主要有两种形式，一种是（　　）线性布局，另一种是（　　）线性布局。

4．创建 Android 程序时，默认使用的布局是（　　）。

5．相对布局中（　　）属性表示“与指定控件右对齐”。

6．帧布局中可以添加多个控件，这些控件会重叠在屏幕（　　）显示。

7．在 Android 控件使用过程中，经常需要根据控件的 ID 获取控件的对象，我们可以使用（　　）方法。

8．在创建控件的时候，可以在布局文件的界面视图中拖曳控件，但本质上还是编辑的（　　）文件。

9．表格布局可以包含多行，（　　）代表是一行。

10．如果界面的某个控件的 id 设置为 btnLogin，那么调用方法 findViewById()时，引用该控件的参数应为（　　）。

二、选择题

1．Android 中有许多控件，这些控件都继承自（　　）类。

A．Control　　B．Window　　C．TextView　　D．View

2．Android 中有许多布局，它们均是用来容纳子控件和子布局的，这些布局均继承自（　　）。

A．Layout　　B．ViewGroup　　C．Container　　D．LinerLayout

3．下列属性中，（　　）属性可以“在指定控件左边”。

A．android:layout_alignleft　　B．android:layout_alignParentLeft

C．android:layout_left　　D．android:layout_toLeftOf

4．相对布局中，“是否跟父布局底部对齐”是属性（　　）。

A．android:layout_alignBottom　　B．android:layout_alignParentBottom

C．android:layout_alignBaseline　　D．android:layout_below

5．以下哪一个控件是用来显示图片？（　　）

A．ImageView　　B．TextView　　C．EditText　　D．Button

6．如果要实现用户单击后触发一定的处理，以下哪一个控件最合适？（　　）

A．ImageView　　B．TextView　　C．EditText　　D．Button

7．以下哪个控件可以用来显示进度？（　　）

A．EditText　　B．ProgressBar　　C．TextView　　D．Button

8．以下哪个方法可以用来获得进度条的当前进度值？（　　）

A．getProgress()　　B．setIndeterminate()
C．setProgress()　　D．incrementProgressBy()

9．ListView 是常用的（　　）类型控件。

A．按钮　　B．图片　　C．列表　　D．下拉列表

10．以下哪个属性用来表示引用图片的资源 id（　　）。

A．text　　B．img　　C．id　　D．src

11．相对布局中，如果指定一个控件位于引用控件的左侧，应该使用（　　）属性。

A．android:layout_toParentLeftOf　　B．android:layout_alignParentLeft
C．android:layout_alignLeft　　D．android:layout_toLeftOf

12．表格布局中 android:layout_column 属性的作用是指定（　　）。

A．行数　　B．列数　　C．总行数　　D．总列数

三、简答题

1．简述五种布局的特点和运用场合。

2．简述本单元所介绍控件的特点和作用。

3．Android 的属性、方法如何使用，它们分别起了什么作用？

单元 4

Android的事件处理与交互实现程序设计

我们开发的 Android 应用程序更多的时候需要与用户进行交互，即对用户的操作进行响应，这就涉及 Android 的事件处理机制，Android 系统给我们提供了两套功能强大的处理机制：① 基于监听的事件处理机制；② 基于回调的事件处理机制。



教学导航

教学目标	（1）理解 Android 的应用组件及其功能、特点 （2）理解并掌握 Android 的 Activity 组件、Intent 组件。 （3）理解并掌握 Android 的事件处理机制，掌握基于监听的事件处理机制和基于回调的事件处理机制使用方法 （4）了解 Android 的对话框与消息框，掌握 Android 的 Toast。
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	8 课时



知识导读

4.1 Android 的应用组件

应用组件是一个 Android 应用程序的基本构建块，Android 应用程序由多个联系松散的应用组件构成，遵守着一个应用程序清单文件 AndroidManifest.xml 的约束，这个清单文件描述了每个组件以及它们如何交互，还有包含了应用程序的硬件和平台需求的元数据(metadata)。以下控件提供了应用程序的基础部分。

（1）Activites（活动）

Activites 是应用程序的表示层，应用程序的每个界面都将是 Activity 类的扩展。Acitvities 用视图（View）构成 GUI 来显示信息、响应用户操作。

一个活动（Activity）表示一个单一的屏幕上的用户界面，就桌面开发而言，一个活动（Activity）相当于一个窗体（Form）。例如，电子邮件应用程序可能有一个活动为显示新的电子邮件列表，另一个活动为撰写电子邮件、阅读电子邮件和其他活动。如果应用程序有一个以上的 Activity，其中一个 Activity 应标记为入口，表明该 Activity 最先被启动。

（2）Services（服务）

Services 是应用程序中的隐形工作者，是在不定的时间运行在后台，不和用户交互的应用

组件。例如播放音乐的时候用户启动了其他 Activity，此时程序要在后台继续播放，而不阻塞用户交互与活动。Services 在后台更新程序的数据源、触发通知（Notification）。在应用程序的 Activities 不激活或不可见时，用于执行依然需要继续的长期处理。

（3）Broadcast Receivers（广播接收器）

Broadcast Receivers 是 Intent 广播的接收器，广播接收器简单地响应其他应用程序或系统的广播消息。如果创建并注册了一个 Broadcast Receiver，应用程序就可以监听匹配了特定过滤标准的广播 Intent。Broadcast Receiver 会自动开启应用程序以响应一个收到的 Intent，使得可以用它们完美地创建事件驱动的应用程序。

Widgets 是可以添加到主屏幕界面（home screen）的可视应用程序控件。作为 Broadcast Receiver 的特殊变种，Widgets 可以为用户创建可嵌入到主屏幕界面的动态的、交互的应用程序控件。

（4）ContentProviders（内容提供者）

ContentProviders 是可共享的数据存储，主要用于对外共享数据，也就是通过 ContentProvider 把应用中的数据共享给其他应用访问，其他应用可以通过 ContentProvider 对指定应用中的数据进行操作。

（5）Intents

Intents 是一个应用程序间（inter-application）的消息传递框架，使用 Intents 可以在系统范围内广播消息或者对一个目标 Activity 或 Service 发送消息，来表示要执行一个动作。系统将辨别出相应要执行活动的目标（target）。

4.2 Activity

1．Activity 概述

Activity 是 Android 组件中最基本也是最为常用的四大组件（Activity，Service、Content Provider、BroadcastReceiver）之一。Activity 是一个应用组件，提供一个屏幕，用户可以通过交互完成某项任务。Activity 中所有操作都与用户密切相关，是一个负责与用户交互的组件，可以通过 setContentView(View)来显示指定控件。

在一个 Android 应用程序中，一个 Activity 通常就是布满整个窗口或者悬浮于其他窗口上的交互界面，它上面可以显示一些控件，也可以监听并处理用户的事件做出响应。Activity 之间通过 Intent 进行通信。

在一个应用程序中通常有多个 Activity，需要在 AndroidManifest.xml 文件中指定程序运行时首先加载的 Activity，即指定一个主 Activity，设置代码如下：

```
<action android:name="android.intent.action.MAIN" />
```

程序第一次运行时用户就会看这个 Activity，这个 Activity 可以通过启动其他的 Activity 进行相关操作。当启动其他的 Activity 时这个当前的 Activity 将会停止，新的 Activity 将会压入栈中，同时获取用户焦点，这时就可在这个 Activity 上操作了。我们知道栈是先进后出的原则，那么当用户按【Back】键时，当前的这个 Activity 销毁，前一个 Activity 重新恢复。

2．Activity 的基本状态及状态转换

在 Android 中，Activity 有四种基本状态：

（1）Active/Runing

一个新 Activity 启动入栈后，它显示在屏幕最前端，此时它处于可见并可和用户交互的激活状态，叫做活动状态或者运行状态（active or running）。

（2）Paused

当 Activity 失去焦点，被一个新的非全屏的 Activity 或者一个透明的 Activity 被放置在栈顶，此时的状态叫做暂停状态（Paused）。此时它依然与窗口管理器保持连接，Activity 依然保持活力（保持所有的状态和成员信息与窗口管理器保持连接），但是在系统内存极端低下的时候将被强行终止掉。所以它仍然可见，但已经失去了焦点，故不可与用户进行交互。

（3）Stoped

如果一个 Activity 被另外的 Activity 完全覆盖掉，叫做停止状态（Stopped）。它依然保持所有状态和成员信息，但是它不再可见，所以它的窗口被隐藏，当系统内存需要被用在其他地方的时候，Stopped 的 Activity 将被强行终止掉。

（4）Killed

如果一个 Activity 是 Paused 或者 Stopped 状态，系统可以将该 Activity 从内存中删除，Android 系统采用两种方式进行删除，要么要求该 Activity 结束，要么直接终止它的进程。当该 Activity 再次显示给用户时，它必须重新开始和重置前面的状态。

当一个 Activity 实例被创建、销毁或者启动另外一个 Activity 时，它在这四种状态之间进行转换，这种转换的发生依赖于用户程序的动作。

Android 程序员可以决定一个 Activity 的“生”，但不能决定它的“死”，也就是说程序员可以启动一个 Activity，但是却不能手动“结束”一个 Activity。

Android 是通过一种 Activity 栈的方式来管理 Activity 的，一个 Activity 的实例的状态决定它在栈中的位置。处于前台的 Activity 总是在栈的顶端，当前台的 Activity 因为异常或其他原因被销毁时，处于栈第二层的 Activity 将被激活，上浮到栈顶。当新的 Activity 启动入栈时，原 Activity 会被压入到栈的第二层。一个 Activity 在栈中的位置变化反映了它在不同状态间的转换。

3．Activity 的生命周期

图 4-1 显示了 Activity 的重要状态转换，矩形框表明 Activity 在状态转换之间的回调接口，开发人员可以重载实现以便执行相关代码，椭圆形表明 Activity 所处的状态。

图 4-1 中，Activity 有三个关键的循环：

（1）整个的生命周期，从 onCreate()开始到 onDestroy()结束

Activity 在 onCreate()设置所有的“全局”状态，在 onDestory()释放所有的资源。例如：某个 Activity 有一个在后台运行的线程，用于从网络下载数据，则该 Activity 可以在 onCreate()中创建线程，在 onDestory()中停止线程。

（2）可见的生命周期，从 onStart()开始到 onStop()结束

在这段时间，可以看到 Activity 在屏幕上，尽管有可能不在前台，不能和用户交互。在这两个接口之间，需要保持显示给用户的 UI 数据和资源等，例如：可以在 onStart 中注册一个 IntentReceiver 来监听数据变化导致 UI 的变动，当不再需要显示时候，可以在 onStop()中注销它。onStart()、onStop()都可以被多次调用，因为 Activity 随时可以在可见和隐藏之间转换。

（3）前台的生命周期，从 onResume()开始到 onPause()结束

在这段时间里，该 Activity 处于所有 Activity 的最前面，和用户进行交互。Activity 可以经常性地在 resumed 和 paused 状态之间切换，例如：当设备准备休眠时；当一个 Activity 处理结果被分发时；当一个新的 Intent 被分发时。所以在这些接口方法中的代码应该属于非常轻量级的。

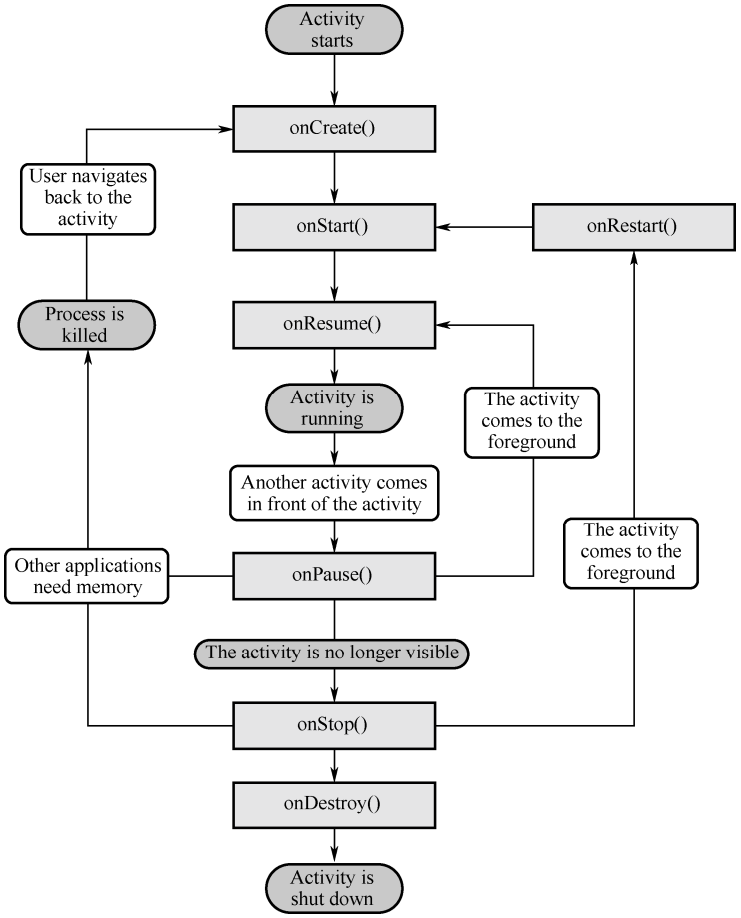


图 4-1　Activity 的重要状态转换示意图

4．Activity 的生命周期方法

Android 提供了很多 Activity 的生命周期方法，常用的方法有以下几个：onCreate()、onStart()、onResume()、onPause()、onStop()、onDestroy()、onRestart()，从名称上就可以粗略看出来这些方法在什么时候执行。

（1）onCreate()

创建 Activity 时调用此方法，用于 Activity 的初始化，还有个 Bundle 类型的参数，可以访问以前存储的状态。在这个方法中需要完成所有的正常静态设置，例如创建一个视图（view）、绑定列表的数据等。调用该方法后一般会调用 onStart()方法。

（2）onRestart()

在 Activity 被停止后重新启动时会调用此方法。其后续会调用 onStart 方法。

（3）onStart()

Activity 在屏幕上对用户可见时调用此方法。

（4）onResume()

Activity 开始和用户交互时调用此方法，这时该 Activity 是在 Activity 栈的顶部，并且接受用户的输入。其后续会调用 onPause()方法。

（5）onPause()

Activity 被暂停时调用此方法，这个方法中通常用来提交一些还没保存的数据更改、停止一些动画或其他一些耗CPU的操作等。无论在该方法里面进行任何操作，都需要较快速完成，因为如果它不返回的话，下一个 Activity 将无法恢复出来。如果 Activity 返回到前台将会调用 onResume()，如果 Activity 变得对用户不可见了将会调用 onStop()。

（6）onStop()

Activity 被停止或者 Activity 变成不可见时调用此方法。可能会因为当前 Activity 正在被销毁，或另一个 Activity（已经存在的 Activity 或新的 Activity）已经恢复了正准备覆盖它，而调用该方法。如果 Activity 正准备返回与用户交互时后续会调用 onRestart()，如果 Activity 正在被释放则会调用 onDestroy()。

（7）onDestroy()

Activity 被从内存中销毁时调用此方法，这是 Activity 能接收到的最后一个调用。可能会因为有人调用了 finish()方法使得当前 Activity 正在关闭，或系统为了保护内存临时释放这个 Activity 的实例，而调用此方法。

Activity 程序入口为方法 onCreate()或 onStart()。一些初始化的操作需要在这两个方法中进行，例如设置 layout、初始化控件、添加事件监听等。每次启动 Activity 都是从 OnCreate()开始，接着执行 onStart()，onResume()。

一个 Activity 在启动的时候会执行 onCreate()→onStart()→onResume()，在结束（或离开）的时候会执行 onPause()→onStop()→onDestroy()，这就是一个 Activity 的生命周期。因此我们要在 onCreate()方法里把 Activity 需要的东西准备好，也就是初始化；在 onResume()方法里对 Activity 里的东西做一些调整；在 onPause()方法里做一些清理和保存工作(保存持久状态)，因为这是最后的机会，因为 onPause()完成之前 Android 不会结束托管 Activity 类的进程，而之后进程可能被结束。

展示 Activity 生命周期及方法应用的 Activity 类代码如表 4-1 所示。

表 4-1　展示 Activity 生命周期及方法应用的 Activity 类代码

序号	代　码
01	public class MainActivity extends Activity {
02	private final static String TAG="ActivityLifeDemo";
03	@Override
04	protected void onCreate(Bundle savedInstanceState) {
05	super.onCreate(savedInstanceState);
06	setContentView(R.layout.activity_main);
07	Log.i(TAG, "onCreate");
08	}

续表

序号	代　码
09	@Override
10	protected void onStart() {
11	Log.i(TAG, "onStart");
12	super.onStart();
13	}
14	@Override
15	protected void onRestart() {
16	super.onRestart();
17	Log.i(TAG, "onRestart");
18	}
19	@Override
20	protected void onResume() {
21	super.onResume();
22	Log.i(TAG, "onResume");
23	}
24	@Override
25	protected void onPause() {
26	super.onPause();
27	Log.i(TAG, "onPause");
28	}
29	@Override
30	protected void onStop() {
31	super.onStop();
32	Log.i(TAG, "onStop");
33	}
34	@Override
35	protected void onDestroy() {
36	super.onDestroy();
37	Log.i(TAG, "onDestroy");
38	}

接下来，我们通过记录操作和查看日志输出信息的方式来看看 Activity 的生命周期过程。运行该程序，可以看到如下日志输出信息：

08-20 18:49:27.428　　1961-1961/…… I/ActivityLifeDemo： onCreate
08-20 18:49:27.428　　1961-1961/…… I/ActivityLifeDemo： onStart
08-20 18:49:27.428　　1961-1961/…… I/ActivityLifeDemo： onResume

按下【Esc】键，可以看到如下日志输出信息：

08-20 18:52:07.347　　1961-1961/…… I/ActivityLifeDemo： onPause
08-20 18:52:13.402　　1961-1961/…… I/ActivityLifeDemo： onStop
08-20 18:52:13.402　　1961-1961/…… I/ActivityLifeDemo： onDestroy

长按【Home】键，弹出最近打开过的应用程序，然后单击 ActivityLifeDemo 程序，可以看到如下日志输出信息：

08-20 18:54:20.299　1961-1961/…… I/ActivityLifeDemo： onCreate
08-20 18:54:20.300　1961-1961/…… I/ActivityLifeDemo： onStart
08-20 18:54:20.301　1961-1961/…… I/ActivityLifeDemo： onResume

按【Home】键，可以看到如下日志输出信息：

08-20 18:54:57.736　1961-1961/…… I/ActivityLifeDemo： onPause
08-20 18:54:58.491　1961-1961/…… I/ActivityLifeDemo： onStop

在应用程序列表中单击 ActivityLifeDemo 程序打开，可以看到如下日志输出信息：

08-20 18:55:43.680　1961-1961/…… I/ActivityLifeDemo： onRestart
08-20 18:55:43.680　1961-1961/…… I/ActivityLifeDemo： onStart
08-20 18:55:43.680　1961-1961/…… I/ActivityLifeDemo： onResume

通过日志输出信息，我们可以看到 Activity 的启动过程：onCreate→onStart→onResume；按下【Esc】返回键时：onPause→onStop→onDestroy，正如上面所说，当按下【Esc】返回键时，此 Activity 弹出栈，程序销毁。我们再次打开时的启动过程又回到 onCreate→onStart→onResume。启动之后按下【Home】键，在 logcat 窗口查看输出信息为：onPause→onStop，再次打开的运行过程为：onRestart→onStart→onResume。

我们通过对 Activity 的各种操作，构成了 Activity 的生命周期，我们看到无论对 Activity 做如何操作，都会接收到相关的回调方法，那么我们在开发的过程中通过调用这些回调方法就可以完成一些操作，例如释放一些重量级的对象、网络连接、数据库连接、文件读取等。

5．Activity 的启动模式

简单地说，启动模式就是 Activity 启动时的策略，在 AndroidManifest.xml 中对 android:launchMode 属性进行设置。

Activity 的启动模式有 4 种，分别为 standard、singleTop、singleTask 和 singleInstance。

在讲解启动模式之前，先分析一下“任务栈”的概念。每个应用都有一个任务栈，是用来存放 Activity 的，功能类似于函数调用的栈，先后顺序代表了 Activity 的出现顺序，例如 Activity1→Activity2→Activity3。

（1）standard 模式

每次激活 Activity 时(startActivity)，都创建 Activity 实例，并放入任务栈。

（2）singleTop 模式

如果某个 Activity 自己激活自己，即任务栈栈顶就是该 Activity，则不需要创建，其余情况都要创建 Activity 实例。

（3）singleTask

如果要激活的那个 Activity 在任务栈中存在该实例，则不需要创建，只需要把此 Activity 放入栈顶，并把该 Activity 以上的 Activity 实例都弹出。

（4）singleInstance

如果应用 1 的任务栈中创建了 MainActivity 实例，如果应用 2 也要激活 MainActivity，则不需要创建，两应用共享该 Activity 实例即可。

6．Android activity 的基本属性

（1）android:allowTaskReparenting

是否允许 activity 更换从属的任务，例如从短信息任务切换到浏览器任务。

（2）android:alwaysRetainTaskState

是否保留状态不变，例如切换回 home，再重新打开。

（3）android:enabled

activity 是否可以被实例化。

（4）android:excludeFromRecents

是否可被显示在最近打开的 activity 列表里。

（5）android:exported

是否允许 activity 被其他程序调用。

（6）android:finishOnTaskLaunch

当用户重新启动这个任务的时候，是否关闭已打开的 activity。

（7）android:launchMode

activity 启动方式，其取值为"standard"、"singleTop"、"singleTask"、"singleInstance"。

（8）android:multiprocess

设置是否允许多进程。

（9）android:name

activity 的类名。

（10）android:process

一个 activity 运行时所在的进程名，所有程序控件运行在应用程序默认的进程中，这个进程名跟应用程序的包名一致。

（11）android:screenOrientation

activity 显示的模式，"unspecified"为默认值，"landscape"为风景画模式，宽度比高度大一些，"portrait"为肖像模式，高度比宽度大。

（12）android:theme

activity 的样式主题，如果没有设置，则 activity 的主题样式从属于应用程序。

7．Activity 的切换

首先简单介绍 Intent（意图）。Intent 类相当于平台中应用程序之间的通信网络，Intent 是一个要执行的操作的抽象说明，相当于各个 Activity 之间的桥梁。在一个 Activity 中想切换到另一个 Activity，需要使用方法 startActivity()，并需要定义一个 Intent 来指定意图的组件类。

（1）最简单的 Activity 切换

定义 Intent，使用 Intent 设置需要切换到哪个 Activity。Activity 的切换一般是通过 Intent 来实现的，Intent 是一个 Activity 到达另一个 Activity 的引路者，它描述了起点（当前 Activity）和终点（目标 Activity）。

使用 Activity 的 startActivity(Intent)方法进行切换的示例代码如下：

```
Intent intent = new Intent();                              //创建一个 Intent 对象
intent.setClass(activity1.this, activity2.class);          //描述起点和目标
startActivity(intent);                                     //开始跳转
```

（2）使用 Intent 传递数据

在切换 Activity 的时候允许传递数据，可以直接使用 Intent 的 putExtra 方法，也可以 new 一个 Bundle 传入。

通常，我们在 Activity 的切换中，希望把前一个 Activity 的某些数据传递给下一个 Activity。这时，我们可以借助 Bundle 来实现。Bundle 相当于数据存储包，用于存放我们想要转达的数据。这里的 Intent 就像一封邮件，里面有送信人地址（原始 Activity），也有收信人地址（目标 Activity），而 Bundle 就是附件也可看作是信件内容。下面是它的实现方法：

① Activity1 发送

```
Intent intent = new Intent();
intent.setClass(activity1.this, activity2.class);     //描述起点和目标
Bundle bundle = new Bundle();                         //创建 Bundle 对象
bundle.putString("something", "Activity1 发来的数据"); //装入数据
intent.putExtras(bundle);                          //把 Bundle 塞入 Intent 里面
startActivity(intent);                                //开始切换
```

② Activity2 接受从 Activity1 发来的数据

```
Intent intent = this.getIntent();          //获取已有的 intent 对象
Bundle bundle = intent.getExtras();        //获取 Intent 里面的 bundle 对象
string = bundle.getString("something");    //获取 Bundle 里面的字符串
```

（3）接受从目标 Activity 返回的处理结果

接受从目标 Activity 返回的处理结果有两种方式方式，一是在 onRestart 方法（onRestrart 表示重启 Activity 时激发的事件）里面接受处理结果，方法实现和 Activity2 接受从 Activity1 发来的数据处理一样。

另一种方式，是采用应答模式切换。应答模式切换和普通的切换不同在于，普通的是有去无回，应答则是有来有往。代码实现也不一样：

① 从 Activity1 切换到 Activity2 调用方法 startActivityForResult(intent,0)；

② 从 Activity2 返回到 Activity1 调用方法 setResult(RESULT_OK, intent)；

③ 接受返回结果调用方法 protected void onActivityResult()。

4.3 Intent

1．Intent 概述

Intent（意图）主要是解决 Android 应用的各项组件之间的通信。Intent 负责对应用中一次操作的动作、动作涉及数据、附加数据进行描述，Android 则根据此 Intent 的描述，负责找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用。因此，Intent 在这里起着一个媒体中介的作用，专门提供组件互相调用的相关信息，实现调用者与被调用者之间的解耦。

Intent 是一种运行时绑定（run-time binding）机制，它能在程序运行过程中连接两个不同的组件。通过 Intent，程序可以向 Android 表达某种请求或者意愿，Android 会根据意愿的内容选择适当的组件来完成请求。例如，有一个 Activity 希望打开网页浏览器查看某一网页的内容，那么这个 Activity 只需要发出 WEB_SEARCH_ACTION 给 Android，Android 就会根据 Intent 的请求内容，查询各组件注册时声明的 IntentFilter，找到网页浏览器的 Activity 来浏览网页。

Android 的三个基本组件（Activity，Service 和 Broadcast Receiver）都是通过 Intent 机制激活的，不同类型的组件有不同的传递 Intent 方式：

① 要激活一个新的 Activity，或者让一个现有的 Activity 做新的操作，可以通过调用 Context.startActivity()或者 Activity.startActivityForResult()方法。

② 要启动一个新的 Service，或者向一个已有的 Service 传递新的指令，调用 Context.startService()方法或者调用 Context.bindService()方法将调用此方法的上下文对象与 Service 绑定。

③ Context.sendBroadcast()、Context.sendOrderBroadcast()、Context.sendStickBroadcast()这 3 个方法可以发送 Broadcast Intent。发送之后，所有已注册的并且拥有与之相匹配 IntentFilter 的 BroadcastReceiver 就会被激活。

Intent 一旦发出，Android 都会准确找到相匹配的一个或多个 Activity，Service 或者 BroadcastReceiver 作响应。所以，不同类型的 Intent 消息不会出现重叠，即 Broadcast 的 Intent 消息只会发送给 BroadcastReceiver，而决不会发送给 Activity 或者 Service。由 startActivity()传递的消息也只会发给 Activity，由 startService()传递的 Intent 只会发送给 Service。

在应用中，我们可以有以下两种形式来使用 Intent：

（1）显式 Intent

显式 Intent 是通过调用 setComponent(ComponentName)或者 setClass(Context, Class)指定了 component 属性的 Intent。通过明确指定具体的组件类，通知应用启动对应的组件。

下面代码方式可以创建显示 Intent 实例化对象，并设定需要传递的参数信息。由于显示 Intent 指定了具体的组件对象，不需要设置 Intent 的其他意图过滤对象。

```
Intent intent = new Intent();
intent.setClass(Context packageContext, Class<?> cls) ;
//内部调用 setComponent(ComponentName)
intent.setClassName(Context packageContext, String className) ;
//内部调用 setComponent(ComponentName)，可以激活外部应用
intent.setClassName(String packageName, String className) ;
intent.setComponent(new ComponentName(this, Class<?> cls));
intent.setComponent(new ComponentName(this, "package Name"));
```

（2）隐式 Intent

隐式 Intent 是没有指定 Component 属性的 Intent，即没有明确指定组件名，系统根据隐式意图中设置的动作(action)、类别(category)、数据 URI 等来匹配最合适的组件。这些 Intent 需要包含足够的信息，这样系统才能根据这些信息，在所有的可用组件中，确定满足此 Intent 的组件。

对于直接 Intent，直接调用就可以了，Android 不需要去做解析，因为目标组件已经很明确，Android 需要解析的是那些间接 Intent，通过解析将 Intent 映射给可以处理此 Intent 的 Activity、Service 或 Broadcast Receiver。

当一个应用要激活另一个应用中的 Activity 时，只能使用隐式 Intent，根据 Activity 配置的意图过滤器创建一个意图，让意图中的各项参数的值都跟过滤器匹配，这样就可以激活其他应用中的 Activity。所以，隐式 Intent 是在应用与应用之间使用的。

2．Intent 的构成

要在不同的 Activity 之间传递数据，就要在 Intent 中包含相应的内容，一般来说数据中最基本的应该包括：

（1）Action

Action 用来指明要实施的动作是什么，例如说 ACTION_VIEW、ACTION_EDIT 等。

一些常用的 Action 如下：

① ACTION_CALL activity：启动一个电话。
② ACTION_EDIT activity：显示用户编辑的数据。
③ ACTION_MAIN activity：作为 Task 中第一个 Activity 启动。
④ ACTION_SYNC activity：同步手机与数据服务器上的数据。
⑤ ACTION_BATTERY_LOW broadcast receiver：电池电量过低警告。
⑥ ACTION_HEADSET_PLUG broadcast receiver：插拔耳机警告。
⑦ ACTION_SCREEN_ON broadcast receiver：屏幕变亮警告。.
⑧ ACTION_TIMEZONE_CHANGED broadcast receiver：改变时区警告。

（2）Data

Data 是执行动作要操作的数据，Android 采用指向数据的一个 URI 来表示，如在联系人应用中，一个指向某联系人的 URI 可能为：content://contacts/1。这种 URI 表示，通过 ContentURI 这个类来描述。

以联系人应用为例，以下是一些 action/data 对，及其他们要表达的意图：

VIEW_ACTION content://contacts/1：显示标识符为"1"的联系人的详细信息。

EDIT_ACTION content://contacts/1：编辑标识符为"1"的联系人的详细信息。

VIEW_ACTION content://contacts/：显示所有联系人的列表。

PICK_ACTION content://contacts/：显示所有联系人的列表，并且允许用户在列表中选择一个联系人，然后把这个联系人返回给父 activity。例如：电子邮件客户端可以使用这个 Intent，要求用户在联系人列表中选择一个联系人。

（3）Category

Category 表示一个类别字符串，包含了有关处理该 Intent 组件的种类信息。例如 LAUNCHER_CATEGORY 表示 Intent 的接受者应该在 Launcher 中作为顶级应用出现，ALTERNATIVE_CATEGORY 表示当前的 Intent 是一系列的可选动作中的一个，这些动作可以在同一块数据上执行。一个 Intent 对象可以有任意个 category，Intent 类定义了许多 category 常数。addCategory()方法为一个 Intent 对象增加一个 category，removeCategory()方法用于删除一个 category，getCategories 方法()用于获取 Intent 所有的 category。

（4）Type

Type 用于显式指定 Intent 的数据类型 MIME（多用途互联网邮件扩展，Multipurpose Internet Mail Extensions 的缩写）。例如，一个组件是可以显示图片数据而不能播放声音文件。一般 Intent 的数据类型能够根据数据本身进行判定，很多情况下，data 类型可在 URI 中找到，例如 content: 开头的 URI，表明数据由设备上的 content provider 提供。但是通过设置这个属性，可以强制采用显式指定的类型而不再进行推导。

（5）component

component 用于指定 Intent 的目标组件的类名称。通常 Android 会根据 Intent 中包含的其他属性的信息，例如 action、data/type、category 进行查找，最终找到一个与之匹配的目标组件。但是，如果 component 这个属性有指定的话，将直接使用它指定的组件，而不再执行上述查找过程。指定了这个属性以后，Intent 的其他所有属性都是可选的。例如：

```
Intent it = new Intent(Activity.Main.this, Activity2.class); startActivity(it);
startActivity(it);
```

（6）extras

extras 是其他所有附加信息的集合，使用 extras 可以为组件提供扩展信息，例如，如果要执行“发送电子邮件”这个动作，可以将电子邮件的标题、正文等保存在 extras 里，传给电子邮件发送组件，Intent 对象有一系列 putXxx()和 setXxx()方法来设定和获取附加信息，这些方法和 Bundle 对象很像。事实上附加信息可以使用 putExtras()和 getExtras()作为 Bundle 来读和写。例如：

```
//用 Bundle 传递数据
Intent it = new Intent(Activity.Main.this, Activity2.class);
Bundle bundle=new Bundle();
bundle.putString("name", "This is from MainActivity!");
it.putExtras(bundle);
startActivity(it); //获得数据
Bundle bundle=getIntent().getExtras();
String name=bundle.getString("name");
```

总之，action、data/type、category 和 extras 一起形成了一种语言，这种语言使系统能够理解诸如“查看某联系人的详细信息”之类的短语。随着应用不断加入到系统中，它们可以添加新的 action、data/type、category 来扩展这种语言。应用也可以提供自己的 Activity 来处理已经存在的这样的“短语”，从而改变这些“短语”的行为。

3．Intent 解析机制

Intent 解析机制主要是通过查找已注册在 AndroidManifest.xml 中的所有<intent-filter>及其中定义的 Intent，最终找到匹配的 Intent。在这个解析过程中，Android 是通过 Intent 的 action、type、category 这三个属性来进行判断的，判断方法如下：

（1）如果 Intent 指明定了 action，则目标组件的 IntentFilter 的 action 列表中就必须包含有这个 action，否则不能匹配。

（2）如果 Intent 没有提供 type，系统将从 data 中得到数据类型。和 action 一样，目标组件的数据类型列表中必须包含 Intent 的数据类型，否则不能匹配。

（3）如果 Intent 中的数据不是 content:类型的 URI，而且 Intent 也没有明确指定 type，将根据 Intent 中数据的 scheme（例如 http:或者 mailto:）进行匹配。Intent 的 scheme 也必须出现在目标组件的 scheme 列表中。

（4）如果 Intent 指定了一个或多个 category，这些类别必须全部出现在组件的类别列表中。例如 Intent 中包含了两个类别：LAUNCHER_CATEGORY 和 ALTERNATIVE_CATEGORY，解析得到的目标组件必须至少包含这两个类别。

Intent-Filter 的属性设置实例如下：

```
    <activity  android:name="NotesList"  android:label="@string/title_notes_
list">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
        <intent-filter>
            <action android:name="android.intent.action.VIEW" />
            <action android:name="android.intent.action.EDIT" />
            <action android:name="android.intent.action.PICK" />
            <category android:name="android.intent.category.DEFAULT" />
            <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
```

```
        </intent-filter>
        <intent-filter>
            <action android:name="android.intent.action.GET_CONTENT" />
            <category android:name="android.intent.category.DEFAULT" />
            <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
        </intent-filter>
    </activity>
```

4．Activity 的 Intent 数据传递

（1）Activity 间的数据传递

① 直接向 Intent 对象中传入键值对，相当于 Intent 对象具有 Map 键值对功能，示例代码如下：

```
intent.putExtra("first", text1.getText().toString());
intent.putExtra("second", text2.getText().toString());
```

② 定义一个 Bundle 对象，在该对象中加入键值对，然后将该对象加入 Intent 中，示例代码如下：

```
Bundle bundle = new Bundle();
bundle.putString("name", "zhang");
bundle.putInt("age", 20);
intent.putExtras(bundle);
```

③ 向 Intent 中添加 ArrayList 集合对象，示例代码如下：

```
intent.putIntegerArrayListExtra(name, value);
intent.putIntegerArrayListExtra(name, value);
```

④ Intent 传递 Object 对象，被传递对象实现 Parcelable 接口，或者实现 Serialitable 接口，示例代码如下：

```
public Intent putExtra(String name, Serializable value)
public Intent putExtra(String name, Parcelable value)
```

（2）Activity 退出时的返回结果

① 通过 startActivityForResult 方式启动一个 Activity。

示例代码如下：

```
MainActivity.this.startActivityForResult(intent, 200);
//200 表示 requestCode 请求码
```

② 新 activity 设定 setResult 方法，通过该方法可以传递 responseCode 和 Intent 对象。

示例代码如下：

```
setResult(101, intent2);    //responseCode 响应码 和 intent 对象
```

③ 在 MainActivity 中覆写 onActivityResult 方法，新 activity 一旦退出，就会执行该方法。

示例代码如下：

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    Toast.makeText(this, data.getStringExtra("info")+"requestCode:"+
    requestCode+"resultCode:"+resultCode, Toast.LENGTH_LONG).show();
}
```

4.4 Android 的事件处理机制

Android 系统给我们提供了两套功能强大的处理机制：① 基于监听的事件处理机制；② 基于回调的事件处理机制。

1．事件

事件是一种用来收集用户与应用程序互动数据的机制，例如按键或触摸屏等放置事件，可以在程序中捕获这些事件，按要求并采取适当的动作。

在 Android 中，事件主要包括点按、长按、拖曳、滑动等，点按又包括单击和双击，另外还包括单指操作和多指操作。所有这些都构成了 Android 中的事件响应。总体来说，所有的事件都由如下三个部分作为基础：按下（ACTION_DOWN）、移动（ACTION_MOVE）和抬起（ACTION_UP）。所有的操作事件首先必须执行的是按下操作（ACTIONDOWN），之后所有的操作都是以按下操作为前提，当按下操作完成后，接下来可能是一段移动（ACTIONMOVE）然后抬起（ACTION_UP），或者是按下操作执行完成后没有移动就直接抬起。这一系列的动作在 Android 中都可以进行控制。

我们知道，所有的事件操作都发生在触摸屏上，而在屏幕上与我们交互的就是各种各样的视图组件（View），在 Android 中，所有的视图都继承于 View，另外通过各种布局组件（ViewGroup）来对 View 进行布局，ViewGroup 也继承于 View。所有的 UI 控件例如 Button、TextView 都是继承于 View，而所有的布局控件例如 RelativeLayout、容器控件例如 ListView 都是继承于 ViewGroup。所以，我们的事件操作主要就是发生在 View 和 ViewGroup 之间。

事件处理程序和事件侦听器如表 4-2 所示。

表 4-2　事件处理程序和事件侦听器

事件处理程序	事件监听器
onClick()	OnClickListener() 当用户单击按钮、文字、图片等控件时被调用，使用 onClick()事件处理程序来处理任何控件的事件
onLongClick()	OnLongClickListener() 当用户单击按钮、文本、图像等控件 1 秒以上时被调用，使用 onLongClick()事件处理程序来处理这样的事件
onFocusChange()	OnFocusChangeListener() 当控件失去焦点时被调用，使用 onFocusChange()事件处理程序来处理这样的事件
onKey()	OnFocusChangeListener() 当用户按下或释放键时被调用，将使用 onKey()事件处理程序来处理这样的事件
onTouch()	OnTouchListener() 当用户按下键及释放键时，或在屏幕上的任何移动手势时被调用，使用 onTouch()事件处理程序来处理这样的事件
onMenuItemClick()	OnMenuItemClickListener() 当用户选择一个菜单项时被调用，使用 onMenuItemClick()事件处理程序来处理这样的事件

2．监听器

监听器是个抽象类，它包含了一个事件触发时系统会去调用的方法，在子类中，可以根据项目的需要重写这个方法。派生后的监听器需要绑定到按钮上，就像一个耳机可以发出声

音，但如果不去戴它，是听不到它发出的声音的。一般的情况是这个按钮可能需要这个监听器，而另外一个按钮需要另外一个监听器，每个监听器各司其职，但功能相似时，也可以多个按钮共同绑定一个监听器。

各种控件都有常用的事件，例如单击按钮，拖动一个滚动条，切换一个 ListView 的选项等，这些控件绑定监听器的方法命名规则是 setOn****Listener。监听器其实是一种回调，它不需要执行 On****Listener，而是系统触发后自动会去调用它。

当用户（也可能系统）触发某个控件的某个事件后，往往要处理一些细节，但它们不具有通用性，例如可能单击按钮后改变它的值，也可能希望单击按钮后弹出一个网页，或者单击按钮后关闭当前 Activity。正由于结果可能太多样，Android 系统设计者索性将实现完全留给 Android 开放者去实现。

可以使用自定义内部类继承监听器抽象类，并实现抽象方法。也可以使用 Java 提供的抽象类的匿名实现。

3．基于监听的事件处理机制

基于监听的事件处理机制模型如图 4-2 所示。

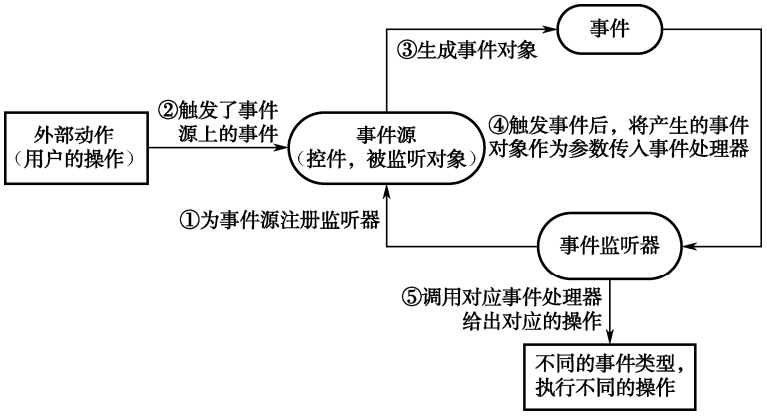


图 4-2　基于监听的事件处理机制模型

事件监听机制是一种委派式的事件处理机制，事件监听机制中由事件源、事件、事件监听器三类对象组成，事件源（控件）将事件处理委托给事件监听器，当事件源发生特定的事件时，就通知事件监听器执行相应的操作。

事件处理模型中，主要涉及三类对象：

① EventSource（事件源）。事件发生的场所，通常就是各个控件，例如按钮、窗口、菜单。

② Event（事件）。事件封装了事件发生的相关信息。

③ EventListener（事件监听器）。监听事件源发生的事件，并对事件作出相应的响应。

事件监听的处理流程如下所示：

① 为某个事件源（控件）设置一个监听器，用于监听用户操作。

② 用户的操作，触发了事件源的监听器。

③ 生成了对应的事件对象。

④ 将这个事件源对象作为参数传给事件监听器。

⑤ 事件监听器对事件对象进行判断，执行对应的事件处理方法。

4．基于回调的事件处理机制

方法回调是将功能定义与功能分开的一种手段，一种解耦合的设计思想，在 Java 中回调是通过接口来实现的。作为一种系统架构，必须要有自己的运行环境，且需要为用户提供实现接口。接口实现依赖于客户，这样就可以达到接口统一，实现不同系统通过在不同的状态下“回调”实现类，从而达到接口和实现的分离。

相比基于监听器的事件处理模型，基于回调的事件处理模型要简单些，该模型中，将事件源和事件监听器是合二为一的，也就是说没有独立的事件监听器存在，即事件源也是事件监听器。当用户在 GUI 控件上触发某事件时，由该控件自身特定的方法负责处理该事件。通常通过重写 Override 控件类的事件处理方法实现事件的处理。

在 Android 中基于回调的事件处理机制使用场景有两个：

（1）自定义 view

当用户在 GUI 控件上激发某个事件时，控件有自己特定的方法会负责处理该事件。通常继承基本的 GUI 控件，且重写该控件的事件处理方法，即自定义 view。

【注意】：在 XML 布局中使用自定义的 view 时，需要使用“全限定类名”。

Android 为 GUI 控件提供了一些事件处理的回调方法，常见 View 控件的回调方法如下：

① 在该控件上触发屏幕事件：boolean onTouchEvent(MotionEvent event);

② 键盘按键按下时触发：boolean onKeyDown(int keyCode,KeyEvent event);

③ 松开键盘按键时触发：boolean onKeyUp(int keyCode,KeyEvent event);

④ 长按控件某个按钮时触发：boolean onKeyLongPress(int keyCode,KeyEvent event);

⑤ 键盘快捷键事件发生时触发：boolean onKeyShortcut(int keyCode,KeyEvent event);

⑥ 当控件的焦点发生改变时触发：protected void onFocusChanged(boolean gainFocus, int direction, Rect previously FocusedRect);

自定义一个 MyButton 类继承 Button 类，然后重写 onKeyDown()方法、onKeyUp()方法、onTouchEvent()方法，其代码如表 4-3 所示。

表 4-3　自定义类 MyButton 的代码

序号	代　　码
01	package com.example.exampleapp0407;
02	import android.widget.Button;
03	import android.view.KeyEvent;
04	import android.view.MotionEvent;
05	import android.util.Log;
06	import android.content.Context;
07	import android.util.AttributeSet;
08	public class MyButton extends Button{
09	private static String TAG = "OK";
10	public MyButton(Context context, AttributeSet attrs) {
11	super(context, attrs);
12	}
13	//重写键盘按钮按下触发的事件
14	@Override

续表

序号	代　码
15	public boolean onKeyDown(int keyCode, KeyEvent event) {
16	super.onKeyDown(keyCode,event);
17	Log.i(TAG, "onKeyDown 方法被调用");
18	return true;
19	}
20	//重写松开键盘按键触发的事件
21	@Override
22	public boolean onKeyUp(int keyCode, KeyEvent event) {
23	super.onKeyUp(keyCode,event);
24	Log.i(TAG,"onKeyUp 方法被调用");
25	return true;
26	}
27	//控件被触摸或单击了
28	@Override
29	public boolean onTouchEvent(MotionEvent event) {
30	super.onTouchEvent(event);
31	Log.i(TAG,"onTouchEvent 方法被调用");
32	return true;
33	}
34	}

MyButton 类中直接重写了 Button 的三个回调方法，当发生单击事件后就不需要我们进行事件监听器的绑定就可以完成方法回调，即控件会处理对应的事件，即事件由事件源（控件）自身处理。

XML 布局文件的代码如下所示。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <com.example.exampleapp0407.MyButton
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="登录" />
</RelativeLayout>
```

程序运行时，logcat 窗口的内容如下所示。

```
08-20 07:39:28.600    1953-1953/com.example.exampleapp0407 I/OK:  onKeyDown方法被调用
08-20 07:39:29.116    1953-1953/com.example.exampleapp0407 I/OK:  onKeyUp方法被调用
08-20 07:39:35.391    1953-1953/com.example.exampleapp0407 I/OK:  onTouchEvent方法被调用
```

（2）基于回调的事件传播

事件处理的先后顺序如下：

① 触发控件绑定的事件监听器。

② 触发控件提供的回调方法。

③ 传播到该控件所在的 Activity 的回调方法。

回调方法的返回值是 boolean 类型，返回值用来标识该回调方法是否已经完全处理该事件了。如果返回值为 false，则说明没处理完，那么就会传播出去，触发控件所在的 Activity 的相关回调方式；如果返回值为 true，就不会继续向外传播了，即后面的回调方法都不会执行。

事件传播的顺序为：监听器优先，然后到 View 组件自身，最后再到 Activity。回调方法的返回值为 false 事件继续传播，返回值为 true 则终止事件传播。

5．实现 Android 事件处理的常见方法

实现 Android 事件处理的常见方法有 4 种，实现这 4 种方法对应的 XML 文件代码如下：

```
<Button
    android:id="@+id/btnLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="登录" />
```

Activity 文件中引入的包如下：

```
import android.widget.Button;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;
```

（1）直接采用匿名内部类作为事件监听器

这种方法将创建一个匿名的执行监听，如果每个类只有一个单控制器，将参数传递给事件处理程序。在这种方法中的事件处理方法可以访问私有数据的活动。

在 setXxxListener 后重写里面的对应方法，例如重写 onClick()方法。这种方法通常都是临时使用一次，复用度不高。类 MainActivity 中实现响应单击事件的主要代码如下：

```
public class MainActivity extendsActivity{
    Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = (Button) findViewById(R.id.btnLogin);
        //绑定匿名的监听器，并执行对应的逻辑代码
        btn.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                //处理 Button 单击事件
                Toast tst = Toast.makeText(MainActivity.this, "单击了登录按钮",
                                    Toast.LENGTH_SHORT);
                tst.show();
            }
        });
    }
}
```

这里没有独立创建 OnClickListener 接口对象，而是直接将实例化对象和设定监听器的处理合二为一，然后调用 setOnClickListener()方法将该监听器与控件进行绑定。

（2）使用自定义的内部类作为事件监听器

这种方法单独定义一个内部类，该类用于实现 OnClickListener 接口对象，并重写 onClick()方法，然后创建该类的实例，最后同样通过 setOnClickListener()方法将监听器与 Button 按钮进行绑定。这种方法可以在 Activity 类中复用事件监听类，也可以直接访问外部类的所有界面控件。

主要代码如下：

```
public class MainActivity extendsActivity{
    Button btn;
    class MyOnClickListener implements OnClickListener
    {
        @Override
        public void onClick(View v) {
            //处理 Button 单击事件
            btn.setText("按钮被单击了");
        }
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = (Button) findViewById(R.id.btnLogin);
        // btn.setOnClickListener(new MyOnClickListener());
        MyOnClickListener clickListener=new MyOnClickListener();
        btn.setOnClickListener(clickListener);
    }
}
```

也可以独立创建 OnClickListener 接口对象，重写该接口中的 onClick()方法，同时创建该接口的监听器对象，然后通过 setOnClickListener()方法将监听器与 Button 按钮进行绑定，这里将实现接口和实例对象的处理合二为一。主要代码如下：

```
public class MainActivity extendsActivity{
    Button btn;
    OnClickListener clickListener=new OnClickListener()
    {
        @Override
        public void onClick(View v) {
            btn.setText("按钮被单击了");
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
       btn = (Button) findViewById(R.id.btnLogin);
        //绑定匿名的监听器，并执行对应的逻辑代码
        btn.setOnClickListener(clickListener);
    }
}
```

还有一种方法就是使用外部类作为事件监听器，就是另外创建一个处理事件的继承于 OnClickListener 的外部类，但由于外部类不能直接访问用户界面类中的控件，要通过构造方法将控件传入使用，代码不够简洁，一般比较少用。

（3）直接使用 Activity 类作为事件监听器

让 Activity 类继承 View.OnClickListener，实现 OnListener 事件监听接口，在 Activity 中

定义重写对应的事件处理器方法，例如重写 onClick(view)方法，当为某控件添加该事件监听器对象时，可以直接 setXxxListener(this)即可。

实现的代码如下：

```
public class MainActivity extendsActivityimplements OnClickListener{
    Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = (Button) findViewById(R.id.btnLogin);
        btn.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        btn.setText("按钮被单击了");
    }
```

这里 Activity 类实现 Listener 接口方法处理主活动，然后调用 setOnClickListener（this）程序。如果应用程序只有一个单一的控件这种方法是很好的，但需要做进一步的编程检查生成的事件（监听器类型）。不能将参数传递到监听器，多个控件时不能起作用。

（4）在 XML 文件中显式指定按钮的 onClick 属性，并把事件绑定到标签

在 XML 布局文件所对应的 Activity 中定义一个事件处理方法，例如 public void myclick(View source)，source 是对应的事件源，接着在布局文件中对应要触发事件的控件设置一个属性，例如 onclick = "myclick"即可。

XML 文件对应的代码如下：

```
    <Button
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="myclick"
        android:text="登录" />
```

实现事件处理对应的代码如下：

```
public class MainActivity extendsActivity{
    Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = (Button) findViewById(R.id.btnLogin);
    }
    // 注意 这里没有@Override 标签
    public void myclick(View v) {
        btn.setText("按钮被单击了");
    }
}
```

在这里事件处理程序 Activity 类没有实现监听器接口，也没有注册任何侦听器方法。相反使用布局文件（activity_main.xml），通过 android:onClick 属性指定的处理程序方法 click 事件。事件处理程序方法必须有一个返回类型为 void，并作为一个参数来检查，方法名称可以是任意的，主类不需要实现任何特定的接口。这种方法不会允许将参数传递给监听器，Android 开

发人员将很难知道哪种方法处理程序控制，需要到 activity_main.xml 文件查看才能知道。其次，不能处理除 click 事件外的任何其他事件。

6．Android Button 监听的常见方式

XML 文件对应的代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/btnLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="登录" />
    <Button
        android:id="@+id/btnCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="取消" />
</LinearLayout>
```

Android button 控件目前主要有如下两种监听方式：

（1）一个 Button 控件对应一个监听

Activity 类对应的主要代码如下：

```
public class MainActivity extendsActivity{
    Button login, cancel;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        login = (Button)findViewById(R.id.btnLogin);
        cancel = (Button)findViewById(R.id.btnCancel);
        login.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                login.setText("登录按钮被单击了");
            }
        });
        cancel.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                cancel.setText("取消按钮被单击了");
            }
        });
    }
}
```

上述这种方法的麻烦在于，如果有很多的 Button，那么就需要对每一个 Button 都写一个类，代码比较复杂。

（2）多个 Button 对应一个监听

可以让 MainActivity 实现接口函数，通过 switch 语句判断 R.id，来对不同的 Button 按钮做出不同的响应。Activity 类对应的主要代码如下：

```
public class MainActivity extendsActivity{
    Button login, cancel;
    class MyClickListener implements OnClickListener {
        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.btnLogin:
                    login.setText("登录按钮被单击了");;
                    break;
                case R.id.btnCancel:
                    cancel.setText("取消按钮被单击了");;
                    break;
                default:
                    break;
            }
        }
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        login = (Button) findViewById(R.id.btnLogin);
        cancel = (Button) findViewById(R.id.btnCancel);
        login.setOnClickListener(new MyClickListener());
        cancel.setOnClickListener(new MyClickListener() );
    }
```

4.5 Android 的对话框与消息框

在 Android 的图形界面中，对话框与消息框是人机交互的重要形式。

1．Android 的 Toast

Android 中的 Toast 是一种简易的消息提示框。和 Dialog 不一样的是，它永远不会获得焦点，无法被用户单击。Toast 显示的时间有限，Toast 会根据用户设置的显示时间后自动消失。Toast 类的最简单的方法是调用静态方法构造所需要的一切，并返回一个新的 Toast 对象。

使用 Toast 类需引入包：

```
import android.widget.Toast;
```

例如，在当前的 activity 中，为某单击事件加入下列代码：

```
Toast.makeText(getApplicationContext(), "提示信息",Toast.LENGTH_SHORT).show();
```

则单击时会出现浮框提示。

（1）创建 Toast 对象

创建 Toast 有如下两种方法：

方法 1：makeText(Context context, int resId, int duration)

参数 context 是 toast 显示在哪个上下文，通常是当前 Activity；resId 指显示内容引用 Resouce 那条数据，就是从 R 类中去指定显示的消息内容；duration 指定显示时间，Toast 默认有 LENGTH_SHORT 和 LENGTH_LONG 两种常量，分别表示短时间显示和长时间显示。

方法 2：makeText(Context context, CharSequence text, int duration)

参数 context 和 duration 与第一个方法相同，参数 text 可以自己写消息内容。

用上面的方法创建 Toast 对象之后调用方法 show()即可显示。

示例代码如下：

```
Toast toast = Toast.makeText(MainActivity.this, "信息内容",
                                                Toast.LENGTH_ LONG);
toast.show();
```

也可写成以下形式：

```
Toast.makeText(MainActivity.this, "信息内容", Toast.LENGTH_LONG).show();
```

（2）设置 Toast 显示位置

设置显示位置有如下两种方法：

方法 1：setGravity(int gravity, int xOffset, int yOffset)，三个参数分别表示起点位置、水平向右位移和垂直向下位移。

方法 2：setMargin(float horizontalMargin, float verticalMargin)，以横向和纵向的百分比设置显示位置，参数均为 float 类型(水平位移正右负左，竖直位移正上负下)。

使用第 1 种方法设置 Toast 的显示位置示例代码如下：

```
toast.setGravity(Gravity.TOP | Gravity.LEFT, 0, 200);
```

使用第 2 种方法设置 Toast 的显示位置示例代码如下：

```
toast.setMargin(-0.5f, 0f);
```

2．Android 的 AlertDialog

AlertDialog 的构造方法全部是 Protected 的，所以不能直接通过 new 来创建出一个 AlertDialog。要创建一个 AlertDialog，就要用到 AlertDialog.Builder 中的 create()方法。

使用 AlertDialog.Builder 创建对话框需要用到以下几个方法：

setTitle()：为对话框设置标题。

setIcon()：为对话框设置图标。

setMessage()：为对话框设置提示内容。

setView()：给对话框设置自定义样式。

setItems()：设置对话框要显示的一个 list，一般用于显示多个命令。

setMultiChoiceItems()：用来设置对话框显示一系列的复选框。

setNeutralButton()：表示普通按钮。

setPositiveButton()：给对话框添加【Yes】按钮。

setNegativeButton()：对话框添加【No】按钮。

create()：创建对话框。

show()：显示对话框。

创建一个简单对话框的代码如下：

```
Dialog alertDialog = new AlertDialog.Builder(this).
                         setTitle("标题").
                         setMessage("对话框的内容").
                         setIcon(R.drawable.ic_launcher).
                         create();
alertDialog.show();
```

也可以写成以下形式：

```
new AlertDialog.Builder(self) .setTitle("标题" .setMessage("对话框的内容")
 .setIcon(R.drawable.ic_launcher).show();
```

这里的代码中新建了一个 AlertDialog，并用 Builder 方法形成了一个对象链，通过一系列的设置方法，构造出我们需要的对话框，然后调用 show 方法显示出来，注意到 Builder 方法的参数 self，这个其实是 Activity 对象的引用，根据所处的上下文来传入相应的引用就可以了。例如在 onCreate 方法中调用，只需传入 this 即可。

创建一个带【确认】和【取消】按钮的对话框代码如下：

```
new AlertDialog.Builder(self) .setTitle("确认") .setMessage("确定吗？")
 .setPositiveButton("是", null) .setNegativeButton("否", null) .show();
```

【注意】：这里的代码包含有两个 null 参数，这里要放的其实是这两个按钮单击的监听程序，由于我们这里不需要监听这些动作，所以传入 null 值简单忽略掉，但是实际开发的时候一般都是需要传入监听器的，用来响应用户的操作。

创建一个可以输入文本的对话框的代码如下：

```
new AlertDialog.Builder(self) .setTitle("请输入")
 .setIcon(android.R.drawable.ic_dialog_info) .setView(new EditText(self))
 .setPositiveButton("确定", null) .setNegativeButton("取消", null)
 .show();
```

这里使用 setView 方法为对话框传入了一个文本编辑框，当然，也可以传入任何的视图对象，例如图片框，WebView 等。

3．Android 中的通知（Notification）

（1）Notification 概述

Notification 是 Android 中常用的一种通知方式，当有未读短信或者未接电话的时候，屏幕的状态栏就会有提示图标，这时可以通过状态栏来读取通知。在使用微信的时候（微信在后台运行），如果有新消息时便会发出声音提示，状态栏也有相应的微信提示。

Notifications 是一个用户通知框架，Notification 让程序不必窃取焦点或中断当前 Activities 就能通知用户。例如，当设备收到一条短消息或一个电话，它会通过闪光灯、发出声音、显示图标或显示消息来进行提醒。也可以在应用程序中使用 Notifications 触发相同的事件。

（2）Android 中 Notification 通知的实现步骤

使用 Notification 时需要导入以下 3 个类：

```
import android.app.PendingIntent;
import android.app.NotificationManager;
import android.app.Notification;
```

Android 中 Notification 通知的实现步骤如下：

① 获取 NotificationManager 对象。NotificationManager 的三个公共方法如下：

cancel(int id)：取消以前显示的一个通知。如果是一个短暂的通知，试图将隐藏；如果是一个持久的通知，将从状态条中移走。

cancelAll()：取消以前显示的所有通知。

notify(int id, Notification notification)：把通知持久地发送到状态条上。

② 初始化 Notification 对象。

③ 设置通知的显示参数。

使用 PendingIntent 来包装通知 Intent，使用 Notification 的 setLatestEventInfo 来设置通知的标题、通知内容等信息。

④ 发送通知。

使用 NotificationManager 的 notify(int id, Notification notification)方法来发送通知。

（3）Notification 的主要属性

Notification 的主要属性如下：

audioStreamType：当声音响起时，所用的音频流的类型。

contentIntent：当通知条目被单击，就执行这个被设置的 Intent。

contentView：当通知被显示在状态条上的时候，同时这个被设置的视图被显示。

defaults：指定哪个值要被设置成默认的。

deleteIntent：当用户单击【Clear All Notifications】按钮删除所有的通知的时候，这个被设置的 Intent 被执行。

icon：状态条所用的图片。

sound：通知的声音。

tickerText：通知被显示在状态条时，所显示的信息。

vibrate：振动模式。

when：通知的时间戳。

【说明】：如果使 Notification 常驻在状态栏，可以把 Notification 的 flags 属性设置为 FLAG_ONGOING_EVENT。

4.6 Android 输出日志信息的方法

开发 Android 程序时，不仅要注意程序代码的正确性与合理性，还要处理程序中可能出现的异常情况，Android SDK 中提供了 Log 类来获取程序运行时的日志信息，还提供了 LogCat 管理器，来查看程序运行的日志信息及错误日志。

Log 类位于 android.util 包中，它继承自 java.lang.Object 类。Log 类提供了一些方法，用来输出日志信息，常用的方法有以下 5 个：Log.v()、Log.d()、Log.i()、Log.w()以及 Log.e()，根据首字母对应 Verbose、Debug、Info、Warning 和 Error。

（1）Log.v 方法的输出内容颜色为黑色，任何消息都会输出。

（2）Log.d 方法的输出内容颜色是蓝色的，仅输出 debug 故障日志信息。

（3）Log.i 方法的输出内容颜色为绿色，输出一般提示性的消息 information，它不会输出 Log.v 和 Log.d 的信息，但会显示 Log.w 和 Log.e 的信息。

（4）Log.w 方法的输出内容颜色为橙色，输出 warning 警告日志信息，一般需要我们注意优化 Android 代码，同时选择它后还会输出 Log.e 的信息。

（5）Log.e 方法的输出内容颜色为红色，输出错误信息，这些错误就需要我们认真分析，查看栈的信息了。

4.7 OnTouchEvent

OnTouchEvent 事件的调用流程比较特别，同时这个事件也是我们实际开发中使用得比较多的。所有的 view 控件都重写了该方法，应用程序可以通过该方法处理手机屏幕的触摸事件。

（1）方法声明

方法声明如下：

```
public boolean onTouchEvent(MotionEvent event);
```

（2）使用流程

① 为某个 view(控件)调用方法 setOnTouchListener(new OnTouchListener())。

② 重写 onTouch()方法。

③ 因为有三种比较常用的触摸状态：

```
MotionEvent.ACTION_DOWN:按下
MotionEvent.ACTION_MOVE:移动
MotionEvent.ACTION_UP:放开
```

所以我们通常是通过 switch(event.getAction)进行分类处理的，然后每个 case 处理对应事件。

（3）方法调用顺序

由方法声明我们知道 onTouchEvent 返回的是一个 boolean 值，其值为 true 或者 false，而这两个的不同之处就是，调用完 onTouch()中的方法后是否再调用外部的方法。

如果我们为某个按钮设置了 onTouch()、onClick()、onLongClick()三个方法，那么它们的调用顺序是怎么样的呢？

如果 onTouchEvent 返回 true：按下时调用 ACTION_DOWN 的方法，调用完后调用 ACTION_MOVE 方法，只要手指一直按着系统就会不断地响应这个方法，原因是 Android 系统对触摸事件比较敏感，虽然我们的手指感觉是静止不动的，其实手指却在不停地微颤抖动。当我们的手指离开了屏幕，这个时候调用 MotionEvent.ACTION_UP 方法。

如果 onTouchEvent 返回 false：和上面那个流程一样，但是离开屏幕后就会调用另外的 2 个方法了：如果是短按，那么会调用 onClick()方法；如果是长按，那么会调用 onLongClick()方法。

4.8 MotionEvent

MotionEvent 是一个触控事件对象，一般在 View 的 onTouchEvent 方法中处理 MotionEvent 对象的，方法原型如下：

```
public boolean onTouchEvent(MotionEvent event)
```

Android 在 MotionEvent 里定义了一系列的手势事件，常见的用户事件类型如下所示。

（1）ACTION_DOWN：用当屏幕检测到第一个触点按下之后就会触发到这个事件。

（2）ACTION_MOVE：当触点在屏幕上移动时触发，触点在屏幕上停留也是会触发的，主要是由于它的灵敏度很高，而我们的手指又不可能完全静止（即使我们感觉不到移动，但其实我们的手指也在不停地抖动）。

（3）ACTION_UP：当最后一个触点松开时被触发。

（4）ACTION_POINTER_DOWN：当屏幕上已经有触点处于按下状态的时候，再有新的触点被按下时触发。

（5）MotionEvent.ACTION_POINTER_UP：当屏幕上有多个点被按住，松开其中一个点时触发（即非最后一个点被放开时）触发。

获取用户屏幕操作的常见方法如下所示。

（1）getX 和 getY 方法：得到用户触摸或单击的位置。

（2）getAction 方法：得到操作事件的类型。

（3）getDownTime 方法：得到用户按下的时间。

（4）getEventTime 方法：得到用户操作的时间。

（5）getPressure 方法：得到用户的触摸压力值。



【任务 4-1】 用户触摸屏幕时显示提示信息

【任务描述】

Android 应用程序运行时，用户触摸屏幕则显示“发生触摸事件”的提示信息，如图 4-3 所示。



图 4-3　用户触摸屏幕时显示的消息框

【知识索引】

（1）Activity 类与 setOnTouchListener()、setBackgroundResource()等方法。

（2）OnTouchListener 接口及 setContentView()、onTouch()等方法。

（3）Toast 类及 maikeText()、show()方法。

【实施过程】

1．创建 Android 项目 App0401 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0401。

2．完善类 MainActivity 与实现程序功能

（1）引入所需的包

本任务相关控件、类与接口涉及的包有多个，引入包的代码如表 4-4 所示。

表 4-4　引入包的代码

序号	代　码
01	import android.app.Activity;
02	import android.os.Bundle;
03	import android.view.Menu;

续表

序号	代　码
04	import android.view.MenuItem;
05	import android.view.MotionEvent;
06	import android.view.View;
07	import android.view.View.OnTouchListener;
08	import android.widget.LinearLayout;
09	import android.widget.Toast;

【说明】：以后各个任务所引入的包不再列出，详见附录 C 有关类和引入包的说明。

（2）在 MainActivity 类中编写与完善方法代码实现程序功能

MainActivity 类继承自父类 Activity，并实现接口 OnTouchListener，首先在 onCreate()方法中，分别调用父类的 onCreate 方法、定义线性布局 layout、调用方法 setOnTouchListener()设置触摸事件监听器、调用方法 setBackgroundResource()设置背景图片、调用方法 setContentView()实现布局；然后定义方法 onTouch()，当触摸事件触发时调用 onTouch()方法，该方法使用 Toast 对象的 maikeText()方法设置要显示的字符串“发生触摸事件”，使用 show()方法显示消息框，主要代码如表 4-5 所示，其他代码为创建项目时自动生成。

表 4-5　onCreate()方法的部分代码

序号	代　码
01	public class MainActivity extendsActivityimplements OnTouchListener{
02	@Override
03	protected void onCreate(Bundle savedInstanceState) {
04	super.onCreate(savedInstanceState);　// 调用父类方法
05	LinearLayout layout = new LinearLayout(this);　// 定义线性布局
06	layout.setOnTouchListener(this);　// 设置触摸事件监听器
07	layout.setBackgroundResource(R.drawable.background);　// 设置背景图片
08	setContentView(layout);　// 实现布局
09	}
10	@Override
11	public boolean onTouch(View v, MotionEvent event) {
12	Toast.makeText(MainActivity.this, "发生触摸事件", Toast.LENGTH_LONG).show();
13	return true;
14	}
15	}

3．程序运行与功能测试

Android 项目 App0401 运行成功时，用户触摸屏幕时会显示如图 4-3 所示消息框。

【任务 4-2】 用户登录时检测用户名的长度合法性

【任务描述】

我们经常会使用用户登录功能，登录时先要输入用户名，通常用户名需要满足一定的要

求，例如用户名长度在 5~9 个字符之间，不得包含特殊字符。设计用户登录界面时，可在用户名文本框的后面添加一个用户名检测按钮，当用户单击此按时检测文本输入框中的内容是否满足输入要求，然后根据判断的结果给出提示信息。



【知识索引】

（1）EditText 控件的 layout_width、layout_height、inputType、ems、id、hint layout_alignParentTop、layout_centerHorizontal 等属性。

（2）Button 控件的 layout_width、layout_height、text、id、layout_below、layout_centerHorizontal、layout_marginTop、onClick 等属性。

（3）findViewById()方法、getText()方法和 setText()方法。



【实施过程】

1．创建 Android 项目 App0402

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0402。

2．完善布局文件 activity_main.xml 与界面设计

先将默认添加的 TextView 控件删除，然后添加 1 个 EditText 控件、1 个 Button 控件和 1 个 TextView 控件，将 EditText 控件的 inputType 属性设置为"textPersonName"、id 属性设置为 etName、hint 属性设置为"请输入用户名"，将 Button 控件的 id 属性设置为 etName、text 属性设置为"注册"、onClick 属性设置为"checkLen"（表示绑定的单击调用方法，当单击【注册】按钮时，调用方法 checkLen 执行其代码），将 TextView 控件的 id 属性设置为 tvInfo、text 属性设置为 TextInfo、layout_marginTop 属性设置为"15dp"。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在主活动 MainActivity 中，首先声明 1 个 TextView 对象和 1 个 EditText 对象，具体代码如下所示：

```
    private TextView tv;       //声明 TextView 对象
    private EditText et;       //声明 EditText 对象
```

（2）在 onCreate()方法中编写代码获取当前布局的控件对象

在 onCreate()方法中，编写代码获取当前布局的控件对象，其他代码为创建项目时自动生成，其代码如表 4-6 所示。

表 4-6　onCreate()方法的代码

序号	代　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 获取当前布局的控件对象
06	et = (EditText)findViewById(R.id.etName);
07	tv = (TextView)findViewById(R.id.tvInfo);
08	}

（3）定义方法 checkLen 实现用户名长度合法性的检测

编写代码定义方法 checkLen()，该方法首先获取用户输入用户名的长度，然后根据用户名长度给出对应的提示信息，其代码如表 4-7 所示。

表 4-7　方法 checkLen()的代码

序号	代　　码
01	public void checkLen(View v){
02	//获取用户输入用户名的长度
03	int len = et.getText().toString().length();
04	//根据输入的用户名的长度，给出对应的提示信息
05	if (len > 5 && len < 9) {
06	tv.setText("用户名合法");
07	}else{
08	tv.setText("用户名长度非法");
09	}
10	}

4．程序运行与功能测试

Android 项目 App0402 运行的初始状态如图 4-4 所示。在文本框中输入 5～9 个合法字符，例如“Android”，然后单击【注册】按钮，则显示“用户名合法”的提示信息，如图 4-5 所示；在文本框中输入的字符大于 9 个，例如“Android Studio”，然后单击【注册】按钮，则显示“用户名长度非法”的提示信息，如图 4-6 所示。



图 4-4　Android 项目 App0402 运行的初始状态



图 4-5　显示“用户名合法”的提示信息

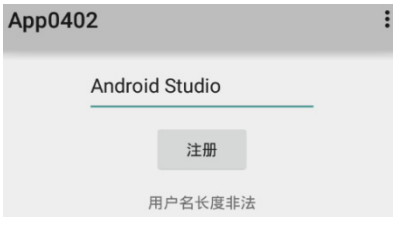


图 4-6　显示“用户名长度非法”的提示信息

【任务 4-3】　获取屏幕单击位置

【任务描述】

随着手机越来越智能，用户与手机交互的手段也越来越多样化，其中最主要的一种形式就是手指触摸屏幕实现的单击和双击操作，如今有很多应用基于屏幕触摸操作来吸引用户。

试编程实现当用户单击屏幕时，显示当前单击位置的屏幕坐标。

【知识索引】

（1）findViewById()方法、onTouchEvent()方法、setText()方法。

（2）MotionEvent 类及常见用户事件类型、getAction()方法、getX()和 getY()方法。

【实施过程】

1．创建 Android 项目 App0403

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0403。

2．完善布局文件 activity_main.xml 与界面设计

将默认添加的 TextView 控件的 id 属性设置为 tv、text 属性设置为“单击屏幕得到相对屏幕位置”，该控件用于显示单击屏幕的坐标。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在主活动 MainActivity 中，首先声明 1 个 TextView 对象，代码为 private TextView tv;

（2）在 onCreate()方法中编写代码获取当前布局的控件对象

在 onCreate()方法中，编写代码获取当前布局的控件对象，其他代码为创建项目时自动生成，其代码如表 4-8 所示。

表 4-8　onCreate()方法的代码

序号	代　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	tv = (TextView)findViewById(R.id.tv);　// 得到当前布局的控件对象
06	}

（3）实现 Activity 的回调方法 onTouchEvent 获取触摸位置的屏幕坐标

编写代码实现回调方法 onTouchEvent()，该方法得到用户单击屏幕的事件，并获取用户触摸屏幕的位置，然后修改 TextView 控件的显示内容，其代码如表 4-9 所示。

表 4-9　回调方法 onTouchEvent()的代码

序号	代　码
01	@Override
02	public boolean onTouchEvent(MotionEvent event) {
03	//当按下屏幕的时候，获取单击位置的 x，y
04	if (MotionEvent.ACTION_DOWN == event.getAction()) {
05	float x = event.getX();
06	float y = event.getY();
07	tv.setText("当前单击的位置是：\nx:"+x+"\ny:"+y);
08	}
09	return super.onTouchEvent(event);
10	}

回调方法 onTouchEvent()中有一个 event 参数，代表用户的操作事件，通过该对象的 getAction()方法可以得到该用户的操作类型，当用户是触摸屏幕的时候，通过方法 getX()和 getY()得到用户触摸或单击的位置，然后修改 TextView 控件的显示内容为屏幕单击的位置信息。

4．程序运行与功能测试

在 Android 项目 App0403 运行状态下，触摸或单击屏幕上任意位置，屏幕上显示对应位置的坐标值，如图 4-7 所示。



图 4-7　屏幕上显示触摸或单击位置的坐标值

【任务 4-4】 实现动态添加联系人

【任务描述】

我们使用手机发送短信时，可以依次添加多个联系人，这就要求当接收到用户某个操作时，能够动态修改当前的页面布局，试编程实现在手机屏幕中动态添加联系人的功能，当用户单击添加联系人按钮时，下方就会添加一个联系人的输入框。

【知识索引】

（1）EditText 类、findViewById()方法、setHint()方法。

（2）OnClickListener 类、setOnClickListener()方法、onClick()方法。

（3）LinearLayout 布局及 addView()、setContentView()、onContentChanged()等方法。

（4）Toast 类及 makeText()、show()方法。

【实施过程】

1．创建 Android 项目 App0404

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0404。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），设置线性布局的 id 为“llAdd”，且将默认添加的 TextView 控件删除，然后添加 1 个 Button 控件，将 Button 控件 text 属性设置为“动态添加联系人”，设置其 id 值为“btnAdd”，方便在 Activity 中取得该对象。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在主活动 MainActivity 中，首先声明多个对象，代码如下所示。

```
private Button btn;
private LinearLayout ll;
private int count = 0;
```

（2）在 onCreate()方法中编写代码获取当前布局的控件对象

在 onCreate()方法中，编写代码获取当前布局的控件对象，其他代码为创建项目时自动生成，其代码如表 4-10 所示。

表 4-10　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	//得到当前布局的控件对象
06	ll = (LinearLayout)findViewById(R.id.llAdd);
07	btn = (Button)findViewById(R.id.btnAdd);
08	setListener();
09	}

表 4-10 中第 06 行得到线性布局对象，第 07 行得到布局中的 Button 对象，方便对布局进行修改。

（3）实现方法 setListener()设置单击事件监听器

编写代码实现方法 setListener()，该方法给 Button 对象设置了单击事件监听器，调用方法 addEditText()添加联系人，setListener()的代码如表 4-11 所示。

表 4-11　方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	//设置 Button 的监听器
03	btn.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View v) {
06	//添加联系人
07	addEditText();
08	}
09	});
10	}

方法 addEditText()用于线性布局中添加 EditText 对象，其代码如表 4-12 所示。

表 4-12　方法 addEditText()的代码

序号	代　　码
01	private void addEditText(){
02	//初始化一个 EditText 对象
03	EditText et = new EditText(MainActivity.this);
04	et.setHint("请输入第"+count+"个联系人的信息！");

续表

序号	代　码
05	//将建立好的 EditText 对象加入 Linearlayout 布局中
06	ll.addView(et);
07	//设置当前页面的布局是 LinearLayout 对象
08	setContentView(ll);
09	}

表 4-12 中第 08 行调用方法 setContentView()给当前的布局设置一个新的布局，用新布局替换理的布局。

（4）实现回调方法 onContentChanged()

当 Activity 调用 setContentView 或者 addContentView 方法时就会修改当前 Activity 的布局，这时 onContentChanged 方法会被自动回调，该方法中显示已有的联系人的数量，并且改变联系人计数变量，其代码代码如表 4-13 所示。

表 4-13　方法 setListener()的代码

序号	代　码
01	@Override
02	public void onContentChanged() {
03	super.onContentChanged();
04	//显示已有的联系人的数量
05	Toast.makeText(MainActivity.this, "已经添加了"+count+"个联系人！",
06	Toast.LENGTH_SHORT).show();
07	count++;
08	}

4．程序运行与功能测试

Android 项目 App0404 的初始运行状态如图 4-8 所示，显示一个 Button 按钮。

单击【动态添加联系人】按钮，手机屏幕下方显示“已经添加了 1 个联系人”的提示信息，如图 4-9 所示，屏幕上方动态添加 1 个 EditText 控件，如图 4-10 所示。



图 4-8　Android 项目 App0404 的初始运行状态



图 4-9　添加 1 个联系人时显示的提示信息

添加 3 个 EditText 控件的屏幕外观如图 4-11 所示。



图 4-10　动态添加第 1 个 EditText 控件

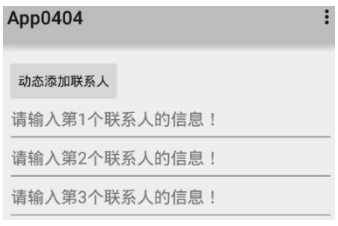


图 4-11　添加 3 个 EditText 控件的屏幕外观

【任务 4-5】 用户注册时检测 Email 格式

【任务描述】

实现用户注册功能时通常需要输入 Email 邮箱，用户注册界面中包含一个 Email 邮箱输入文本框和显示检测邮箱格式结果的图片，当输入的邮箱格式符合相应的规则时，显示带“正确”标识的图片，否则显示带“错误”标识的图片。

【知识索引】

（1）EditText 控件 inputType、layout_width、layout_height、ems、id、layout_alignParentTop、layout_toEndOf、selectAllOnFocus 等属性。

（2）OnKeyListener 类、setOnKeyListener()方法、onKey()方法、getAction()方法。

（3）View 类、KeyEvent 类及常见的用户事件类型。

（4）getText()方法、matches()方法、setImageResource()方法。

【实施过程】

1．创建 Android 项目 App0405 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0405，将本任务所需的图片 right.png 和 wrong.png 导入或复制到 res/drawable 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

先将默认添加的 TextView 控件删除，然后添加 1 个 TextView 控件、1 个 EditText 控件和 1 个 ImageView 控件，将 TextView 控件的 id 属性设置为 tvInfo、text 属性设置为“请输入 Email：”，将 EditText 控件 inputType 属性设置为“textEmailAddress”、id 值设置为“etEmail”、selectAllOnFocus 属性设置为“true”，将 ImageView 控件的 id 属性设置为 ivStatus。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在主活动 MainActivity 中，首先声明多个对象，代码如下所示。

```
private EditText et = null;
private ImageView iv = null;
```

（2）在 onCreate()方法中编写代码获取当前布局的控件对象

在 onCreate()方法中，编写代码获取当前布局的控件对象，其他代码为创建项目时自动生成，其代码如表 4-14 所示。

表 4-14　onCreate()方法的代码

序号	代　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 获得文本框控件
06	et = (EditText) findViewById(R.id.etEmail);
07	// 获得图片控件
08	iv = (ImageView) findViewById(R.id.ivStatus);
09	setListener();
10	}

表 4-14 中第 06 行得到布局中的 EditText 对象，第 08 行得到布局中的 ImageView 对象。

（3）实现方法 setListener()设置单击事件监听器

编写代码实现方法 setListener()，该方法给 EditText 对象设置了键盘事件监听器，在 EditText 控件中每次输入字符时都会回调 onKey 方法，setListener()的代码如表 4-15 所示。

表 4-15　方法 setListener()的代码

序号	代　码
01	private void setListener() {
02	// 为文本框控件绑定键盘事件
03	et.setOnKeyListener(new View.OnKeyListener() {
04	@Override
05	public boolean onKey(View v, int keyCode, KeyEvent event) {
06	switch (event.getAction()) { // 得到操作类型
07	case KeyEvent.ACTION_DOWN: //键盘按下
08	String strInput = et.getText().toString(); //获得文本框的内容
09	if (strInput.matches("\\w+@\\w+\\.\\w+")) { //验证通过
10	iv.setImageResource(R.drawable.right);
11	}else {
12	iv.setImageResource(R.drawable.wrong);
13	}
14	break;
15	case KeyEvent.ACTION_UP: //键盘弹起
16	break;
17	}
18	return false;
19	}
20	});
21	}

4．程序运行与功能测试

Android 项目 App0405 的初始运行状态如图 4-12 所示。

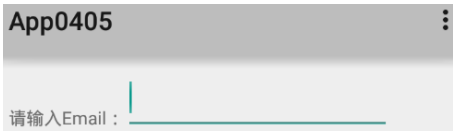


图 4-12　Android 项目 App0405 的初始运行状态

在 Email 文本框中输入“better”，由于不符合 Email 格式规则，显示错误标识图片，如图 4-13 所示。

在 Email 文本框中输入完整的 Email“better@163.com”，由于符合 Email 格式规则，显示正确标识图片，如图 4-14 所示。



图 4-13　显示“错误”标识图片

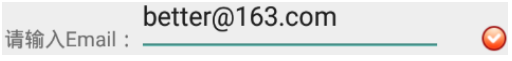


图 4-14　显示“正确”标识图片

【任务 4-6】 打开浏览器浏览网页

【任务描述】

在文本框中输入网址，然后单击【打开】按钮，即可打开浏览器显示相应的网页，并显示用户输入的网址，编程实现这一功能。

【知识索引】

（1）OnClickListener 类、setOnClickListener()方法、onClick()方法。

（2）String 类、getText()方法、toString()方法、equals()方法。

（3）Uri 类、parse()方法。

（4）Intent 类、startActivity()方法、ACTION_VIEW 常量。

（5）Toast 类、makeText()方法、show()方法、LENGTH_SHORT 常量。

【实施过程】

1．创建 Android 项目 App0406

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0406。

2．完善布局文件 activity_main.xml 与界面设计

将默认添加的 TextView 控件删除，然后添加 1 个 EditText 控件和 1 个 Button 控件，将 EditText 控件的 id 属性设置为 etURL、hint 属性设置为“请输入网址”、text 属性设置为“www.baidu.com”，将 Button 控件的 id 属性设置为“btnOpen”、hint 属性设置为“浏览网页”。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明 Button 对象和 EditText 对象，代码如下所示。

```
private Button btn1;
private EditText et1;
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中得到 EditText 对象和 Button 对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 4-16 所示。

表 4-16　onCreate()方法的代码

序号	代　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	et1 = (EditText)findViewById(R.id.etURL);　//得到布局中的 EditText 对象
06	btn1 = (Button)findViewById(R.id.btnOpen);　//得到布局中的 Button 对象
07	setListener();　//设置对象的监听器
08	}

（3）编写代码为按钮自定义单击监听器

编写代码自定义单击监听器 OnClickListener，方法 setListener()实现代码如表 4-17 所示，

当用户单击按钮时首先获取用户在 EditText 中输入的内容，如果为空则使用 Toast 提示用户输入网址，如果不为空则将用户输入的网址构成 Uri 对象，接着定义 intent 对象，设置 Action 为 Intent.ACTION_VIEW，通过 Intent.ACTION_VIEW 来显示此 Uri 的内容，最后调用 startActivity 方法打开对应的 Activity。

表 4-17　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	//设置 btn 的单击监听器
03	btn1.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View v) {
06	//得到用户输入的网站地址
07	String url = et1.getText().toString();
08	//当用户输入不为空时
09	if (!"".equals(url)) {
10	//在用户输入的地址前加上 http://,然后使用处理后的网址构成 Uri 对象
11	Uri uri = Uri.parse("http://"+url);
12	//定义 intent 对象，通过 Intent.ACTION_VIEW 来显示此 Uri 的内容
13	Intent it　= new Intent(Intent.ACTION_VIEW,uri);
14	//启动 Activity
15	startActivity(it);
16	}
17	else{
18	//如果用户输入的 url 为空的话，使用 Toast 提示用户
19	Toast.makeText(MainActivity.this,"请输入网址...",
20	Toast.LENGTH_SHORT).show();
21	}
22	}
23	});
24	}

4．程序运行与功能测试

Android 项目 App0406 的初始运行状态如图 4-15 所示，单击【浏览网页】按钮，弹出如图 4-16 所示的网页，且显示所输入的网址。



图 4-15　项目 App0406 的初始运行状态



图 4-16　打开浏览网页



单元小结

Android 系统给我们提供了两套功能强大的处理机制：① 基于监听的事件处理机制；② 基于回调的事件处理机制。基于监听的事件处理机制简单来说就是为事件源（控件）绑定特定的监听器，当用户触发了事件后，交给监听器去处理，根据不同的事件执行不同的操作。基于监听的事件处理是一种委派式（Delegation）的事件处理方式。UI 控件（事件源）将发生的事件委派给特定的对象（监听器）处理，这种委派式的事件处理方式将事件源和监听器分离，能提高程序的可维护性。基于回调的事件处理机制就是重写 UI 控件或者 Activity 的回调方法，基于回调的事件处理代码简洁，但不能访问外围类的资源。而基于监听的事件处理，可以把监听器作为内部类而存在，这样就可以访问外围类的资源了。

本单元主要介绍 Android 的应用组件及其功能、Android 的对话框与消息框，重点探析了 Activity 组件、Intent 组件和 Android 的事件处理机制，学会编写应用程序时灵活应用 Activity 组件和 Intent 组件，学会基于监听的事件处理机制和基于回调的事件处理机制使用方法。



单元习题

一、填空题

1. Android 的四大组件分别是（　　）、（　　）、（　　）和 ContentProvider。

2．Activity 生命周期的三种状态分别是（　　）、（　　）和（　　）。

3．Activity 的 4 种启动模式分别是（　　）、（　　）、（　　）和 singleInstance。

4．Activity 中 Intent 寻找目标控件的方式有两种，分别为（　　）和（　　）。

5．Activity 生命周期中“回到前台，再次可见时执行”时调用的方法是（　　）。

6．Android 程序提供了一个（　　）方法来实现回传数据。在数据传递时，如果需要获取返回的数据，需要使用（　　）方法。

7．Intent 可以用来开启 Activity，同样它也可以用来在（　　）之间传递数据。

8．创建 Toast 使用 makeText 方法的第一个参数代表 Activity 的（　　）。

9．Toast 的作用是用于（　　）。

10．LogCat 区域中有 v、d．i、w 和 e 等 5 个字母，其中 v 代表（　　），d 代表（　　），i 代表（　　），w 代表（　　），e 代表（　　）。

二、选择题

1．如果需要捕捉某个控件的事件，需要为该控件创建（　　）。

A．属性　　B．方法　　C．监听器　　D．事件

2．一个 Android 应用程序默认会包含（　　）个 Activity。

A．1　　B．5　　C．10　　D．若干

3．下列方法中，Activity 从启动到关闭不会执行的是（　　）。

A．onCreate()　　B．onStart()　　C．onResume()　　D．onRestart()

4．下列控件中，不能使用 Intent 启动的是（　　）。

A．Activity　　B．启动服务　　C．广播　　D．内容提供者

5．startActivityForResult 方法接收两个参数，第 1 个是 Intent，第 2 个是（　　）。

A．resultCode　　B．requestCode　　C．请求码　　D．data

6．下列关于 Activity 的描述中，错误的是（　　）。

A．Activity 是 Android 的四大控件之一

B．Activity 有 4 种启动模式

C．Activity 通常用于开启一个广播事件

D．Activity 就像一个界面管理员，用户在界面上的操作是通过 Activity 来管理的

7．Android 中下列属于 Intent 的作用的是（　　）。

A．实现应用程序间的数据共享

B．可以保持应用在后台运行，而不会因为切换页面而消失

C．可以实现界面间的切换，可以包含动作和动作数据，连接四大控件的纽带

D．处理一个应用程序整体性的工作

8．在 Activity 的生命周期中，当 Activity 处于栈顶时，此时处于哪种状态？（　　）

A．活动　　B．暂停　　C．停止　　D．销毁

9．在 Activity 的生命周期中，当 Activity 被某个 AlertDialog 覆盖掉一部分之后，会处于哪种状态？（　　）

A．活动　　B．暂停　　C．停止　　D．销毁

10．Action 属性 ACTION_DIAL 代表（　　）标准动作。

A．显示电话拨号面板　　B．显示直接打电话的界面

C．向用户显示数据　　D．提供编辑数据的途径

11．如果需要显示 id 为 1 的联系人信息，Intent 中的 Action 属性与 Data 属性应该设定为什么？（　　）。

A．ACTION_VIEW content://contacts/people/1

B．ACTION_DIAL content://contacts/people/1

C．ACITON_EDIT content://contacts/people/1

D．ACTION_CALL content://contacts/people/1

12．Toast 创建完毕后，需要显示出来此时需要调用以下哪个方法？（　　）

A．makeText　　B．show　　C．create　　D．view

13．以下哪个类对应 Android 中的提示对话框？（　　）

A．AlertDialog　　B．Dialog　　C．ShowDialog　　D．Alert

14．Android 中有一个服务用来管理通知，它是（　　）。

A．Service　　B．NotificationManager

C．Notice　　D．DialogBuilder

三、简答题

1．简要说明 Activity 的 4 种启动模式的区别。

2．简要说明 Activity 的 3 种状态以及不同状态使用的方法。

3．Android 提供了多种提示信息方式，简述它们各自的优缺点？

4．Intent 有哪些重要属性，Activity 之间是如何进行信息传递的？

5．Android 的监听器如何使用，有什么作用？

单元 5

Android 的数据存储与数据共享程序设计

对于一个实用的 Android 应用程序，数据存储操作是必不可少的。Android 系统共提供了四种数据存储方式。分别是：SharePreference、SQLite、ContentProvider 和 File。由于 Android 系统中，数据基本都是私有的，要实现数据共享，最佳的方式是使用 ContentProvider。



教学导航

教学目标	（1）熟悉 Android 系统提供的四种数据存储方式：SharePreference、SQLite、ContentProvider 和 File （2）学会编写程序使用 SQLite 数据库存储数据 （3）学会编写程序使用 SharedPreference 对象存储数据 （4）学会编写程序使用 ContentProvider 对象存储数据 （5）学会编写程序使用 File 对象存储数据
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	8 课时



知识导读

5.1 Android 系统的数据存储方式简介

Android 系统提供了四种数据存储方式，分别是：SharePreference、SQLite、Content Provider 和 File。

（1）SQLite：SQLite 是一个轻量级的数据库，支持基本 SQL 语法，是常被采用的一种数据存储方式。Android 为此数据库提供了一个名为 SQLiteDatabase 的类，封装了一些操作数据库的 API。

（2）SharedPreference：除 SQLite 数据库外，SharedPreference 是另一种常用的数据存储方式，其本质就是一个 XML 文件，常用于存储较简单的参数设置。

（3）ContentProvider：Android 系统中能实现所有应用程序共享的一种数据存储方式，由于数据在各应用间通常是私密的，所以此存储方式较少使用，但是其又是必不可少的一种存储方式，例如音频，视频，图片和通讯录，一般都可以采用此种方式进行存储。每个

ContentProvider 都会对外提供一个公共的 URI（包装成 Uri 对象），如果应用程序有数据需要共享时，就需要使用 ContentProvider 为这些数据定义一个 URI，然后其他的应用程序就通过 ContentProvider 传入这个 URI 来对数据进行操作。URI 由“content://”、数据的路径、标识 ID(可选) 3 个部分组成。

（4）File：即常说的文件（I/O）存储方法，常用于存储大数量的数据。

5.2 使用 SQLite 数据库存储数据

SQLite 是一个非常流行的嵌入式数据库，它支持 SQL 语言，并且只利用很少的内存就有很好的性能。它是开源的，许多开源项目（(Mozilla、PHP、Python）都使用了 SQLite。SQLite 由 SQL 编译器、内核、后端以及附件组成。SQLite 通过利用虚拟机和虚拟数据库引擎（VDBE）技术，使调试、修改和扩展 SQLite 的内核变得更加方便。

Android 在运行时（run-time）集成了 SQLite，所以每个 Android 应用程序都可以使用 SQLite 数据库。由于 JDBC 会消耗太多的系统资源，所以 JDBC 对于手机这种内存受限设备来说并不合适。因此，Android 提供了一些新的 API 来使用 SQLite 数据库，Android 开发中，程序员需要学会使用这些 API。

1．SQLite 的数据类型

SQLite 和其他数据库最大的不同就是对数据类型的支持，创建一个表时，可以在 Create Table 语句中指定某列的数据类型，并且可以把任何数据类型放入任何列中。当某个值插入数据库时，SQLite 将检查它的类型，如果该类型与关联的列不匹配，则 SQLite 会尝试将该值转换成该列的类型。如果不能转换，则该值将作为其本身具有的类型存储。

SQLite 是一种轻型数据库，只有五种数据类型，分别是 NULL（空值）、Integer（整数）、Real（浮点数）、Text（字符串）和 Blob（大数据）。在 SQLite 中，并没有专门设计 Boolean 和 Date 类型，因为 Boolean 型可以用 Integer 的 0 和 1 代替 true 和 false，而 Date 类型则可以拥有特定格式的 Text、Real 和 Integer 的值来代替显示，为了能方便地操作 Date 类型，SQLite 提供了一组函数。这样简单的数据类型设计更加符合嵌入式设备的要求。

2．SQLite 数据库操作的主要方法

在 Android 系统中提供了 android.database.sqlite 包，用于进行 SQLite 数据库的增、删、改、查操作。其主要方法如下：

① beginTransaction()：开始一个事务。

② close()：关闭连接，释放资源。

③ delete(String table, String whereClause, String[] whereArgs)：根据给定条件，删除符合条件的记录。

④ endTransaction()：结束一个事务。

⑤ execSQL(String sql)：执行给定 SQL 语句。

⑥ insert(String table, String nullColumnHack, ContentValues values)：根据给定条件，插入一条记录。

⑦ openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)：根据给定条件连接数据库，如果此数据库不存在则创建一个数据库。

⑧ query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy,

String having, String orderBy)：执行查询。

⑨ rawQuery(String sql, String[] selectionArgs)：根据给定 SQL，执行查询。

⑩ update(String table, ContentValues values, String whereClause, String[] whereArgs)：根据给定条件，修改符合条件的记录。

3．创建 SQLite 数据库

可以通过 openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)方法创建数据库。另外还可以通过写一个继承 SQLiteOpenHelper 类的方式创建数据库。

4．创建数据表与插入数据。

Android 系统并没有提供特别的创建数据表的方法，数据表通过 SQL 语句创建，代码如下：

```
    db.execSQL("create table tab(_id INTEGER PRIMARY KEY AUTOINCREMENT, name
TEXT NOT NULL)");
```

数据表创建好之后，通过 insert(String table, String nullColumnHack, ContentValues values)方法插入数据。其中参数含义分别为：table 为目标表名；nullColumnHack 指定表中的某列列名。因为在 SQLite 中，不允许插入所有列均为 null 的记录，因此如果初始值为空时，此列需显式赋予 null。

5．修改数据

update(String table, ContentValues values, String whereClause, String[] whereArgs)方法用于修改数据，其四个参数的具体含义如下：table 为目标表名；values 为要被修改成为的新值；whereClause 为 where 子句，其中可带“？”占位符，如没有子句，则为 null；whereArgs 用于替代 whereClause 参数中“？”占位符的参数，如不需传入参数，则为 null。

6．查询数据

可以使用 Android 提供的 query()和 rowQuery()方法执行查询操作，query()和 rawQuery()方法两者的不同在于所需参数的不同。rawQuery 方法需要开发者手动写出查询 SQL，而 query 方法是由目标表名、where 子句、order by 子句、having 子句等诸多子句由系统组成 SQL 语句。两方法都返回 Cursor 对象，使用完毕后调用方法 close()关闭。

7．删除数据

删除数据只需要调用 delete 方法，传入参数即可。方法 delete(String table, String whereClause, String[] whereArgs)的三个参数的具体含义如下：table 为目标表名；whereClause 为 where 子句，其中可带“？”占位符，如没有子句，则为 null；whereArgs 用于替代 whereClause 参数中“？”占位符的参数，如不需传入参数，则为 null。

示例代码如下：

```
    db.delete("tab", "_id=? or name=?", new String[]{"8", "Tom"});
```

5.3 使用 SharedPreference 对象存储数据

除了 SQLite 数据库外，SharedPreferences 也是一种轻量级的数据存储方式，它的本质是基于 XML 文件存储 key-value 键值对数据，通常用来存储一些配置信息。可以通过 edit()方法来修改其内容，通过 Commit()方法来提交修改后的内容。

（1）实现 SharedPreferences 存储的步骤

SharedPreferences 对象本身只能获取数据而不支持存储和修改，存储修改是通过 Editor 对象实现。实现 SharedPreferences 存储的步骤如下：

① 根据 Context 获取 SharedPreferences 对象。

② 利用 edit()方法获取 Editor 对象。

③ 通过 Editor 对象存储 key-value 键值对数据。

④ 通过 commit()方法提交数据。

SharedPreferences 是一个接口，无法创建 SharedPreferences 实例，可以通过 Context.getSharedPreferences(String name.int mode)来得到一个 SharedPreferences 实例。其中 name 是指文件名称，不需要加后缀.xml，系统会自动添加上；mode 是指定读写方式，其值有三种，分别为 Context.MODE_PRIVATE（指定该 SharedPreferences 数据只能被本应用程序读、写）、Context.MODE_WORLD_READABLE（指定该 SharedPreferences 数据能被其他应用程序读，但不能写）、Context.MODE_WORLD_WRITEABLE（指定该 SharedPreferences 数据能被其他应用程序读写）。

（2）SharedPreferences 常用的方法

SharedPreferences 对象与 SQLite 数据库相比，免去了创建数据库、创建表、编写 SQL 语句等诸多操作，相对而言更加方便、简洁。但是 SharedPreferences 也有其自身缺陷，只能存储 boolean、int、float、long 和 String 五种简单的数据类型，无法进行条件查询。所以不论 SharedPreferences 的数据存储操作是如何简单，它也只能是存储方式的一种补充，而无法完全替代如 SQLite 数据库这样的数据存储方式。

SharedPreferences 接口主要负责读取应用程序的 Preferences 数据，它提供了如下常用方法来访问 SharedPreferences 的 key_value 键值对。

① public abstract boolean contains (String key)：判断 SharedPreferences 是否包含特定 key 的数据。

② edit()：为 preferences 创建一个编辑器 Editor，通过创建的 Editor 可以修改 preferences 的数据，但必须执行 commit()方法。返回一个 Edit 对象用于操作 SharedPreferences。

③ getAll()：获取 SharedPreferences 里全部的 key-value 数据对。

④ getXXX(String key , XXX defvlaue)：获取 SharedPreferences 中指定 key 所对应的 value。如果该 key 不存在，返回默认值 defvalue，其中 XXX 可以是 boolean、float、int、long、String 等基本类型的值。

（3）Editor 接口常用的方法

由于 SharedPreference 是一个接口，而且在这个接口里并没有提供写入数据和读取数据的方法。但是在其内部有一个 Editor 接口，Edit 这个接口有一系列的方法用于操作 SharedPreference。Editor 接口的常用方法如下：

① public abstract SharedPreferences.Editor clear()：清空 SharedPreferences 里所有的数据。

② public abstract boolean commit()：当 Editor 编辑完成后，调用该方法可以提交修改，而且必须要调用这个数据才修改。

③ public abstract SharedPreferences.Editor putXXX (String key, boolean XXX)：向 SharedPreferences 存入指定的 key 对应的数据，其中 XXX 可以是 boolean、float、int、long、

String 等基本类型的值。

④ public abstract SharedPreferences.Editor remove (String key)：删除 SharedPreferences 里指定 key 对应的数据项。

5.4 使用 ContentProvider 存储数据

在 Android 中，ContentProvider 是一种数据包装器，适合在不同进程间实现信息的共享。例如，在 Android 中 SQLite 数据库是一个典型的数据源，我们可以把它封装到 ContentProvider 中，这样就可以很好地为其他应用提供信息共享服务。其他应用在访问 ContentProvider 时，可以使用一组 URI 方式进行数据操作，简化了读写信息的复杂度。

ContentProvider 是一个抽象类，ContentProvider 是 Android 平台中，在不同应用程序之间实现数据共享的一种机制。一个应用程序如果需要让别的程序可以操作自己的数据，即可采用这种机制。并且此种方式忽略了底层的数据存储实现，ContentProvider 提供了一种统一的通过 Uri 实现数据操作的方式。其步骤为：

① 在当前应用程序中定义一个 ContentProvider。

② 在当前应用程序的 AndroidManifest.xml 中注册此 ContentProvider。

③ 其他应用程序通过 ContentResolver 和 Uri 来获取 ContentProvider 的数据。

外界的程序通过 ContentResolver 接口可以访问 ContentProvider 提供的数据，在 Activity 中通过 getContentResolver()可以得到当前应用的 ContentResolver 实例。ContentResolver 提供的接口和 ContentProvider 中需要实现的接口对应。

在 Android 中，ContentProvider 是数据对外的接口，程序通过 ContentProvider 访问数据而不需要关心数据具体的存储及访问过程，这样既提高了数据的访问效率，同时也保护了数据。Activity 类中有一个继承自 ContentWapper 的 getContentResolver()无参数方法，该方法返回一个 ContentResolver 对象，通过调用其 query()、insert()、update()、delete()方法访问数据，实现对 ContentProvider 中数据的存取操作。这种存储方式相比 SQLite 和 SharedPreferences，其复杂性是显而易见的，但是在处处可见“云”的今天，程序间的数据交互需求令 ContentProvider 存储机制变成必不可少的。

5.5 使用 File 对象存储数据

在 Android 中，使用 Files 对象存储数据主要有两种方式，一种是 Java 提供的 I/O 流体系，即使用 FileOutputStream 类提供的 openFileOutput()方法和 FileInputStream 类提供的 openFileInput()方法访问磁盘上的文件内容；另一种是使用 Environment 类的 getExternalStorageDirectory()方法对 Android 模拟器的 SD 卡进行数据读写。

使用 Java 提供的 I/O 流体系可以很方便地对 Android 模拟器本地存储的数据进行读写操作，其中 FileOutputStream 类的 openFileOutput()方法用于打开相应的输出流；而 FileInputStream 类的 openFileInput()方法用于打开相应的输入流。默认情况下，使用 I/O 流保存的文件仅对当前应用程序可见，对于其他应用程序是不可见的，即不能访问其中的数据。如果用户卸载了该应用程序，则保存数据的文件也会一起被删除。

每个 Android 设备都支持共享的外部存储用来保存文件，可以是 SD 卡可以移除的存储介

质，也可以是手机内存等不可以移除的存储介质。保存在外部存储的文件是全局可读的，而且在用户使用 USB 连接电脑后，可以修改这些文件。在 Android 程序中，对 SD 卡等外部存储的文件进行操作时，需要使用 Environment 类的 getExternalStorageDirectory()方法，该方法用于获取外部存储器（SD 卡）的文件夹。

5.6 URI 及其组成

URI 是一个通用资源标志符，将其分为 4 个部分：

① 无法改变的标准前缀，包括；“content://”、“tel://”等。当前缀是“content://”时，说明通过一个 ContentProvider 控制这些数据。

② URI 的标识，它通过 authorities 属性声明，用于定义了是哪个 ContentProvider 提供这些数据。对于第三方应用程序，为了保证 URI 标识的唯一性，它必须是一个完整的、小写的类名。例如;“content://com.test.data.myprovider”。

③ 路径，可以近似理解为需要操作的数据库中表的名字，例如：“content://hx.android.text.myprovider/name”中的 name。

④ 如果 URI 中包含需要获取记录的 id，则返回该 id 对应的数据，如果没有 id，就表示返回全部。



【任务 5-1】设计可记住用户名和密码的登录界面

【任务描述】

设计可记住用户名和密码的登录界面，即用户界面文本框中输入用户名和密码，然后单击【登录】按钮即可登录，在登录之前如果用户选择了“自动登录”复选框，则下次打开此界面时，程序会自动填写上次所输入的用户名和密码，直接单击【登录】按钮即可登录。

【知识索引】

（1）LinearLayout 布局及其嵌套、TextView、EditText、CheckBox 和 Button 控件。

（2）SharedPreferences 对象及 getSharedPreferences()方法。

（3）Editor 对象及 edit()、putBoolean()、putString()、commit()等方法。

（4）Intent 及 startActivity()方法。

（5）Toast 类及 makeText()方法、show()方法。

【实施过程】

1．创建 Android 项目 App0501

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0501。

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加多个字符串定义，strings.xml 文件的代码如表 5-1 所示。

表 5-1　字符串定义文件 strings.xml 的代码

序号	布局代码
01	<resources>
02	<string name="app_name">App0501</string>
03	<string name="action_settings">Settings</string>
04	<string name="tv_username">用户名：</string>
05	<string name="tv_password">密\u0020\u0020\u0020\u0020 码：</string>
06	<string name="cb_autologin">自动登录</string>
07	<string name="btn_login">登录</string>
08	<string name="tv_success">登录成功！</string>
09	</resources>

3．完善布局文件 activity_main.xml 与界面设计

将界面的整体布局方式设置为线性布局，并将默认添加的 TextView 控件删除，然后分别添加 3 个嵌套的线性布局，分别用于设置用户名输入行、密码输入行和登录按钮行。在用户名输入行和密码输入行各添加 1 个 TextView 控件和 1 个 EditText 控件，在登录按钮行添加 1 个 CheckBox 和 1 个 Button 控件，并设置好各个控件的属性，调整好每一行控件的相对位置。修改完善后布局文件 activity_main.xml 的代码如表 5-2 所示。

表 5-2　布局文件 activity_main.xml 的代码

序号	布 局 代 码
01	<LinearLayout ……
02	android:orientation="vertical" >
03	<!-- 用户名输入行 -->
04	<LinearLayout
05	……>
06	<TextView
07	……
08	android:text="@string/tv_username" />
09	<EditText
10	android:id="@+id/et_username"
11	……
12	android:layout_weight="1"
13	android:inputType="text"
14	android:text="admin"> />
15	</EditText>
16	</LinearLayout>
17	<!-- 密码输入行 -->
18	<LinearLayout
19	…… >
20	<TextView
21	……

续表

序号	布 局 代 码
22	android:text="@string/tv_password"
23	android:textColor="@android:color/black"
24	android:textSize="20sp" />
25	<EditText
26	android:id="@+id/et_password"
27	……
28	android:layout_width="0dip"
29	android:layout_weight="1"
30	android:inputType="textPassword"
31	android:textColor="@android:color/black" />
32	</LinearLayout>
33	<!-- 登录按钮行 -->
34	<LinearLayout
35	……
36	android:orientation="horizontal" >
37	<CheckBox
38	android:id="@+id/cb_autologin"
39	……
40	android:layout_weight="600"
41	android:text="@string/cb_autologin"
42	android:checked="true"/>
43	<Button
44	android:id="@+id/btn_login"
45	……
46	android:layout_weight="1"
47	android:layout_marginLeft="20dp"
48	android:text="@string/btn_login" />
49	</LinearLayout>
50	</LinearLayout>

4．新添加布局文件 success_main.xml 与界面设计

新添加 1 个布局文件 success_main.xml，用于显示登录时的提示信息，其界面仅包括 1 个 TextView 控件，设置该控件的 text 属性为 tv_success。

5．完善类 MainActivity 与实现程序功能

（1）声明对象

在类 MainActivity 中，首先声明多个必要的对象，其代码如表 5-3 所示。

表 5-3　声明对象的代码

序号	代　　码
01	private EditText userName;　// 用户名
02	private EditText password;　// 密码

续表

序号	代　　码
03	private CheckBox autologin;　// 是否自动登录复选框
04	private Button btnLogin;　　// 登录按钮
05	// 声明一个 SharedPreferences 用于保存数据
06	private SharedPreferences spSettings = null;
07	private static final String PREFS_NAME = "NamePwd";

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取界面中控件对象，调用方法 setListener()绑定控件事件，调用方法 getData()获得数据，其代码如表 5-4 所示。

表 5-4　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	userName = (EditText)findViewById(R.id.et_username);
06	password = (EditText)findViewById(R.id.et_password);
07	autologin = (CheckBox)findViewById(R.id.cb_autologin);
08	btnLogin = (Button) findViewById(R.id.btn_login);
09	// 绑定控件事件
10	setListener();
11	// 获取数据
12	getData();
13	}

（3）定义方法 setListener()为登录按钮绑定单击事件监听器

方法 setListener()的代码如表 5-5 所示，该方法实现为登录按钮绑定单击事件监听器。

表 5-5　方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	// 为登录按钮绑定事件
03	btnLogin.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View arg0) {
06	// 判断用户名和密码
07	if ("admin".equals(userName.getText().toString())
08	&& "123456".equals(password.getText().toString())) {
09	// 判断复选框是否选中
10	if (autologin.isChecked()) {
11	spSettings = getSharedPreferences(PREFS_NAME,MODE_PRIVATE);

续表

序号	代　　码
12	// 得到 Editor 对象
13	Editor edit = spSettings.edit();
14	// 记录保存标记
15	edit.putBoolean("isKeep", true);
16	// 记录用户名
17	edit.putString("username", userName.getText().toString());
18	// 记录密码
19	edit.putString("password", password.getText().toString());
20	edit.commit(); // 提交
21	} else {
22	spSettings = getSharedPreferences(PREFS_NAME,MODE_PRIVATE);
23	// 得到 Editor 对象
24	Editor edit = spSettings.edit();
25	// 记录保存标记
26	edit.putBoolean("isKeep", false);
27	// 记录用户名
28	edit.putString("username", "");
29	// 记录密码
30	edit.putString("password", "");
31	edit.commit(); // 提交
32	}
33	// 跳转到首页
34	Intent intent = new Intent(MainActivity.this,SuccessActivity.class);
35	startActivity(intent);
36	} else {
37	// 显示错误提示
38	Toast.makeText(getApplicationContext(), "用户名或密码错误",
39	Toast.LENGTH_SHORT).show();
40	}
41	}
42	});
43	}

第 07 和 08 行判断用户名和密码是否正确，这里只进行了固定值的判断。第 11 行得到系统的 SharedPreferences 对象，第 13 行得到对应的 Editor 对象，然后写入用户名和密码数据。第 34 行跳转到 SuccessActivity 页面。

【注意】：SharedPreferences 中的数据是以文件的形式存储的，只不过 Android 系统封装了这些文件存储的过程和方法，对应的文件内容是使用 XML 格式来存储的。

（4）定义方法 onResume()在界面显示数据之前获取存储的数据

方法 onResume()的代码如表 5-6 所示，该方法用于在界面显示数据之前得到之前存储的数据。

表 5-6　方法 onResume()的代码

序号	代　　码
01	@Override
02	protected void onResume() {
03	// 在界面显示数据之前得到之前存储的数据
04	super.onResume();
05	getData();
06	}

（5）定义方法 getData()获取存储的数据

方法 getData()的代码如表 5-7 所示，该方法用于获取存储的数据。

表 5-7　方法 getData()的代码

序号	代　　码
01	private void getData() {
02	// 得到 sharedpreferences 对象
03	spSettings = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
04	// 判断是否之前存储过用户名密码
05	if (spSettings.getBoolean("isKeep", false)) {
06	// 如果之前存储过，则显示在相应文本框内
07	userName.setText(spSettings.getString("username", ""));
08	password.setText(spSettings.getString("password", ""));
09	} else {
10	// 否则显示空
11	userName.setText("");
12	password.setText("");
13	}
14	}

6．定义类 SuccessActivity 与实现程序功能

定义一个类 SuccessActivity，在其方法 onCreate 中调用 setContentView 方法加载布局文件 success_main，其代码比较简单，如表 5-8 所示。

表 5-8　类 SuccessActivity 的代码

序号	代　　码
01	package com.example.app0501;
02	import android.app.Activity;
03	import android.os.Bundle;
04	public class SuccessActivity extends Activity {
05	@Override
06	protected void onCreate(Bundle savedInstanceState) {
07	super.onCreate(savedInstanceState);
08	setContentView(R.layout.success_main);

续表

序号	代　　码
09	}
10	}

7．在 AndroidManifest.xml 文件中注册 Activity

类 SuccessActivity 定义完成后，必须在 AndroidManifest.xml 文件编写代码进行注册，否则无法打开，注册 Activity 的代码如表 5-9 中第 17 行所示。

表 5-9　AndroidManifest.xml 文件的代码

序号	代　　码
01	<?xml version="1.0" encoding="utf-8"?>
02	<manifest xmlns:android="http://schemas.android.com/apk/res/android"
03	package="com.example.app0501" >
04	<application
05	android:allowBackup="true"
06	android:icon="@mipmap/ic_launcher"
07	android:label="@string/app_name"
08	android:theme="@style/AppTheme" >
09	<activity
10	android:name=".MainActivity"
11	android:label="@string/app_name" >
12	<intent-filter>
13	<action android:name="android.intent.action.MAIN" />
14	<category android:name="android.intent.category.LAUNCHER" />
15	</intent-filter>
16	</activity>
17	<activity android:name=".SuccessActivity"></activity>
18	</application>
19	</manifest>

8．程序运行与功能测试

Android 项目 App0501 的运行成功时，登录一个登录界面，在“用户名”文本框中输入正确的用户名，这里输入“admin”，在“密码”文本框中输入对应的密码，这里输入“123456”，选中复选框“自动登录”，如图 5-1 所示，然后单击【登录】按钮，弹出另一个界面显示“登录成功”提示信息，如图 5-2 所示。

由于前一次登录时已将复选框“自动登录”选中，再一次运行该程序时，登录界面会自动填写上一次输入的用户名和密码，如图 5-1 所示。



图 5-1　在用户登录界面输入正确的用户名和密码



图 5-2　“登录成功”提示信息

【任务 5-2】对 Android 模拟器中的 SD 卡进行操作

【任务描述】

编写程序在界面中显示 SD 卡的路径，并在 SD 卡上创建图片文件 img01.png。

【知识索引】

（1）LinearLayout 布局、TextView 控件。

（2）Environment 类及 getExternalStorageState()方法。

（3）File 类及 exists()、createNewFile()、canWrite()、getPath()方法。

【实施过程】

1．创建 Android 项目 App0502

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0502。

2．完善布局文件 activity_main.xml 与界面设计

将界面的布局方式设置为线性布局，并将默认添加的 TextView 控件删除，然后添加 2 个 TextView 控件，设置 TextView 控件的 id 属性分别为 tvPath、tvInfo。

3．完善类 MainActivity 与实现程序功能

在 onCreate()方法中使用 getExternalStorageDirectory()方法获得 SD 卡根路径，然后使用 createNewFile()方法创建图片文件并给出提示信息，其代码如表 5-10 所示。

表 5-10　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	TextView path=(TextView)findViewById(R.id.tvPath);
06	TextView info=(TextView)findViewById(R.id.tvInfo);
07	// 取得 SD 卡当前的状态
08	String strSD = Environment.getExternalStorageState();
09	//如果当前系统有 SD 卡存在
10	if (strSD.equals(Environment.MEDIA_MOUNTED)) {
11	File rootPath = Environment.getExternalStorageDirectory();// 获得 SD 卡根路径
12	path.setText("SD 卡的路径为: " + rootPath.getPath());
13	if(rootPath.exists() && rootPath.canWrite()){
14	File file = new File(rootPath, "img01.png");
15	try {
16	if (file.createNewFile()) {
17	info.setText(file.getName() + "文件创建成功！");
18	} else {
19	info.setText(file.getName() + "文件创建失败！");
20	}
21	} catch (IOException e) {

续表

序号	代　　码
22	e.printStackTrace();
23	}
24	}else {
25	info.setText("SD 卡不存在或者不可写！");
26	}
27	}
28	}

4．修改 AndroidManifest.xml 配置文件增加外部存储写入权限

在 AndroidManifest.xml 配置文件添加如下代码，增加外部存储写入权限。

```
<uses-sdk android:minSdkVersion="17" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

5．程序运行与功能测试

Android 项目 App0502 的运行结果如图 5-3 所示。

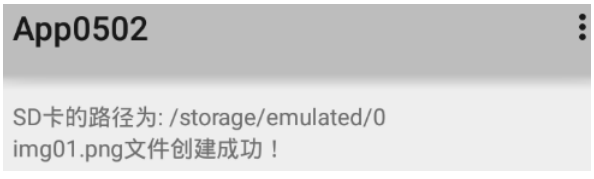


图 5-3　Android 项目 App0502 的运行结果

【提示】：如果文件创建成功后，再一次运行程序创建同一个文件，则会出现“文件创建失败!”的提示信息，其原因是在同一个位置创建名称相同的文件导致文件创建失败。

【任务 5-3】使用 SharedPreferences 实现 Activity 之间的数据传递

【任务描述】

编写程序使用 SharedPreferences 保存用户输入的用户名和密码，并在第 2 个 Activity 的界面显示用户名和密码信息。

【知识索引】

（1）FileOutputStream 类以及 openFileOutput()、openFileInput ()、write()、read(()、flush()、close()等方法。

（2）FileNotFoundException 类和 IOException 类。

（3）Intent 类以及 setClass)、startActivity()方法。

【实施过程】

1．创建 Android 项目 App0503

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0503。

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加多个字符串定义，strings.xml 文件的代

码如表 5-11 所示。

表 5-11　字符串定义文件 strings.xml 的代码

序号	布 局 代 码
01	<resources>
02	<string name="app_name">App0503</string>
03	<string name="action_settings">Settings</string>
04	<string name="usernameInfo">用户名：</string>
05	<string name="passwordInfo">密\u0020\u0020\u0020\u0020 码：</string>
06	<string name="loginInfo">登录</string>
07	</resources>

3．完善布局文件 activity_main.xml 与界面设计

参考【任务 5-1】修改完善后的布局文件 activity_main.xml。

4．新添加布局文件 result.xml 与界面设计

新添加 1 个布局文件 result.xml，用于显示用户登录时输入的用户名和密码，其界面仅包括 2 个 TextView 控件，设置 TextView 控件的 id 属性分别为 tvUsername2、tvPassword2。

5．完善类 MainActivity 与实现程序功能

（1）声明对象

在类 MainActivity 中，首先声明多个必要的对象，其代码如下所示：

```
private EditText username ;    // 获得用户名控件
private EditText password ;    // 获得密码控件
private Button login ;         // 获得按钮控件
String strName;                // 获得用户名
String strPassword;            // 获得密码
```

（2）在 MainActivity 类的 onCreate()方法中编写代码实现程序功能

在 MainActivity 类的 onCreate()方法中获取用户登录时输入的用户名和密码，然后将其保存到 SharedPreferences 类中，最后使用 Intent 跳转到 SharedPreferencesReadActivity，其代码如表 5-12 所示。

表 5-12　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);　　//设置对应的 XML 布局文件
05	username = (EditText) findViewById(R.id.etUsername);　　// 获得用户名控件
06	password = (EditText) findViewById(R.id.etPassword);　　// 获得密码控件
07	login = (Button) findViewById(R.id.btnLogin);　　// 获得按钮控件
08	login.setOnClickListener(new View.OnClickListener() {
09	@Override

续表

序号	代　　码
10	public void onClick(View v) {
11	strName = username.getText().toString();　　// 获得用户名
12	strPassword = password.getText().toString();　　// 获得密码
13	FileOutputStream fos = null;
14	try {
15	fos = openFileOutput("LOGIN", MODE_PRIVATE);　// 获得文件输出流
16	fos.write((strName + " " + strPassword).getBytes());　// 保存用户名和密码
17	fos.flush();　// 清除缓存
18	} catch (FileNotFoundException e) {
19	e.printStackTrace();
20	} catch (IOException e) {
21	e.printStackTrace();
22	} finally {
23	if (fos != null) {
24	try {
25	fos.close();　// 关闭文件输出流
26	} catch (IOException e) {
27	e.printStackTrace();
28	}
29	}
30	}
31	Intent intent = new Intent();// 创建 Intent 对象
32	intent.setClass(MainActivity.this, ReadActivity.class);
33	startActivity(intent);// 实现跳转
34	}
35	});
36	}

6．定义类 ReadActivity 与实现程序功能

定义一个类 ReadActivity，在其方法 onCreate 中从 SharedPreferences 中读取已保存的用户名和密码，然后使用 TextView 控件显示，方法 onCreate 的代码如表 5-13 所示，该方法调用了自定义方法 setLoginInfo()。

表 5-13　类 ReadActivity 中方法 onCreate 的代码

序号	代　　码
01	protected void onCreate(Bundle savedInstanceState) {
02	super.onCreate(savedInstanceState);　　// 调用父类方法
03	setContentView(R.layout.result);　　// 使用布局文件
04	setLoginInfo();
05	}

自定义方法 setLoginInfo()的代码如表 5-14 所示。

表 5-14　方法 onCreate 的代码

序号	代　　码
01	private void setLoginInfo() {
02	TextView username,password ;
03	String data,strUsername,strPassword ;
04	FileInputStream fis = null;
05	byte[] buffer = null;
06	try {
07	fis = openFileInput("LOGIN");　　// 获得文件输入流
08	buffer = new byte[fis.available()];　　// 定义保存数据的数组
09	fis.read(buffer);　　// 从输入流中读取数据
10	} catch (FileNotFoundException e) {
11	e.printStackTrace();
12	} catch (IOException e) {
13	e.printStackTrace();
14	} finally {
15	if (fis != null) {
16	try {
17	fis.close();　　//关闭文件输入流
18	} catch (IOException e) {
19	e.printStackTrace();
20	}
21	}
22	}
23	username = (TextView) findViewById(R.id.tvUsername2);
24	password = (TextView) findViewById(R.id.tvPassword2);
25	data = new String(buffer);　　// 获得数组中保存的数据
26	strUsername = data.split(" ")[0];　　// 获得 username
27	strPassword = data.split(" ")[1];　　// 获得 password
28	username.setText("用户名：" + strUsername);　　// 显示用户名
29	password.setText("密　　码：" + strPassword);　// 显示密码
30	}

7．在 AndroidManifest.xml 文件中注册 2 个 Activity 并配置启动项

类 ReadActivity 定义完成后，必须在 AndroidManifest.xml 文件编写代码进行注册，否则无法打开，另外还需要配置启动项，代码如表 5-15 中第 17 行所示。

表 5-15　AndroidManifest.xml 文件的代码

序号	代　　码
01	<?xml version="1.0" encoding="utf-8"?>
02	<manifest xmlns:android="http://schemas.android.com/apk/res/android"
03	package="com.example.app0503" >
04	<application

续表

序号	代　　码
05	android:allowBackup="true"
06	android:icon="@mipmap/ic_launcher"
07	android:label="@string/app_name"
08	android:theme="@style/AppTheme" >
09	<activity
10	android:name=".MainActivity"
11	android:label="@string/app_name" >
12	<intent-filter>
13	<action android:name="android.intent.action.MAIN" />
14	<category android:name="android.intent.category.LAUNCHER" />
15	</intent-filter>
16	</activity>
17	<activity android:name=".ReadActivity" />
18	</application>
19	</manifest>

8．程序运行与功能测试

Android 项目 App0503 的运行成功时，登录一个登录界面，在“用户名”文本框中输入正确的用户名，这里输入“admin”，在“密码”文本框中输入对应的密码，这里输入“123456”，如图 5-4 所示，然后单击【登录】按钮，弹出另一个界面显示用户登录时输入的用户名和密码，如图 5-5 所示。



图 5-4　在用户登录界面输入正确的用户名和密码



图 5-5　显示用户登录输入的用户名和密码

【任务 5-4】使用 SQLite 数据库保存用户输入的用户名和密码

【任务描述】

Android 提供了内置的 SQLite 数据库来存储数据，SQLite 使用 SQL 命令提供了完整的关系型数据库处理能力，使用 SQLite 数据库保存用户输入的用户名和密码，并在第 2 个 Activity 中显示，编写程序实现这一功能。

【知识索引】

（1）SQLiteDatabase 类及 execSQL()、insert()、query()等方法。

（2）SQLiteOpenHelper 类及 onCreate()、getWritableDatabase()等方法。

（3）ContentValues 类及 put()方法。

（4）Cursor 类及 getCount()、moveToFirst()、close()等方法。

（5）Intent 类以及 setClass()、startActivity()方法。



【实施过程】

1．创建 Android 项目 App0504

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0504。

2．在 strings.xml 文件中定义字符串

打开字符串定义文件 strings.xml，在该文件中添加多个字符串定义，strings.xml 文件的代码如表 5-1 所示。

3．完善布局文件 activity_main.xml 与界面设计

参考【任务 5-1】完善布局文件 activity_main.xml。

4．新添加布局文件 result.xml 与界面设计

新添加 1 个布局文件 result.xml，用于显示用户登录时输入的用户名和密码，其界面仅包括 2 个 TextView 控件。

5．创建用户类 User

在 com.example.app0504 包中创建文件夹 util，然后在该文件夹中新建 User 类，用来封装用户写入的用户名、密码等登录信息，其代码如表 5-16 所示。

表 5-16　User 类的代码

序号	代　　码
01	public class User {
02	private int id;　　// 保存用户的 ID
03	private String strUsername;　// 保存用户名
04	private String strPassword;　// 保存密码
05	public User() {
06	}
07	public User(String username, String　password) {
08	this.strUsername = username;
09	this.strPassword = password;
10	}
11	public void setUsername(String username) {
12	this.strUsername = username;
13	}
14	public void setPassword(String password) {
15	this.strPassword = password;
16	}
17	public int getId() {
18	return id;
19	}
20	public String getUsername() {
21	return strUsername;
22	}

续表

序号	代　　码
23	public String getPassword() {
24	return strPassword;
25	}
26	}

6．创建用户类 DBHelper

在 com.example.app0504.util 包中新建 DBHelper 类，该类定义了多个方法实现数据库操作，其中 insert()方法用于向数据表中保存数据，query()方法用于根据 ID 值查询数据，其代码如表 5-17 所示。

表 5-17　DBHelper 类的代码

序号	代　　码
01	import android.content.ContentValues;
02	import android.content.Context;
03	import android.database.Cursor;
04	import android.database.sqlite.SQLiteDatabase;
05	import android.database.sqlite.SQLiteOpenHelper;
06	
07	public class DBHelper {
08	private static final String DATABASE_NAME = "datastorage";　// 保存数据库名称
09	private static final int DATABASE_VERSION = 1;　// 保存数据库版本号
10	private static final String TABLE_NAME = "users";　// 保存数据表名称
11	private static final String ID = "_id";　// 保存 ID 值
12	private static final String USERNAME = "username";　// 保存用户名
13	private static final String PASSWORD = "password";　// 保存密码
14	private DBOpenHelper helper;
15	private SQLiteDatabase db;
16	private static class DBOpenHelper extends SQLiteOpenHelper {
17	private static final String CREATE_TABLE = "create table "
18	+ TABLE_NAME + " (" + ID + " integer primary key autoincrement, "
19	+ USERNAME + " text not null, "
20	+ PASSWORD + " text not null);";　// 定义创建表格的 SQL 语句
21	public DBOpenHelper(Context context) {
22	super(context, DATABASE_NAME, null, DATABASE_VERSION);
23	}
24	@Override
25	public void onCreate(SQLiteDatabase db) {
26	db.execSQL(CREATE_TABLE);// 创建表格
27	}
28	@Override
29	public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

续表

序号	代　　码
30	db.execSQL("drop table if exists " + TABLE_NAME);　// 删除旧版数据表
31	onCreate(db);　　// 创建数据表
32	}
33	}
34	public DBHelper(Context context) {
35	helper = new DBOpenHelper(context);　// 创建 SQLiteOpenHelper 对象
36	db = helper.getWritableDatabase();　　// 获得可写的数据库
37	}
38	public void insert(User user) {　　// 向数据表中插入数据
39	ContentValues values = new ContentValues();
40	values.put(USERNAME, user.getUsername());
41	values.put(PASSWORD, user.getPassword());
42	db.insert(TABLE_NAME, null, values);
43	}
44	public User query(int id) {　　// 根据 ID 值查询数据
45	User user = new User();
46	Cursor cursor = db.query(TABLE_NAME, new String[] { USERNAME, PASSWORD },
47	"_id = " + id, null, null, null, null);
48	if (cursor.getCount() > 0) {　　// 如果获得的查询记录条数大于 0
49	cursor.moveToFirst() ;　　// 将游标移动到第一条记录
50	user.setUsername(cursor.getString(0));　　// 获得用户名的值然后进行设置
51	user.setPassword(cursor.getString(1));　　// 获得密码的值然后进行设置
52	return user;
53	}
54	cursor.close();　　// 关闭游标
55	return null;
56	}
57	}

7．完善类 MainActivity 与实现程序功能

在 MainActivity 类的 onCreate()方法中获取用户登录时输入的用户名和密码，然后将其保存到 SQLite 数据库中，最后使用 Intent 跳转到 SQLiteReadActivity，其代码如表 5-18 所示。

表 5-18　MainActivity 类 onCreate()方法的代码

序号	代　　码
01	@Override
02	public void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);　　// 调用父类方法
04	setContentView(R.layout.activity_main);　// 应用自定义布局文件
05	final EditText username = (EditText) findViewById(R.id.etUsername);　// 获得用户名控件
06	final EditText password = (EditText) findViewById(R.id.etPassword);　// 获得密码控件
07	Button login = (Button) findViewById(R.id.btnLogin);　　// 获得按钮控件

续表

序号	代　　码
08	login.setOnClickListener(new View.OnClickListener() {
09	@Override
10	public void onClick(View v) {
11	String strUsername = username.getText().toString();　// 获得用户名
12	String strPassword = password.getText().toString();　// 获得密码
13	User user = new User(strUsername, strPassword);
14	DBHelper helper = new DBHelper(SQLiteWriteActivity.this);
15	helper.insert(user);　// 向数据表中插入数据
16	Intent intent = new Intent();　// 创建 Intent 对象
17	// 指定跳转到 SQLiteReadActivity
18	intent.setClass(SQLiteWriteActivity.this, SQLiteReadActivity.class);
19	startActivity(intent);　// 实现跳转
20	}
21	});
22	}

8．定义类 SQLiteReadActivity 与实现程序功能

定义一个类 SQLiteReadActivity，在其方法 onCreate 从 SQLite 数据库中读取已保存的用户名和密码，然后使用 TextView 控件显示，方法 onCreate 的代码如表 5-19 所示。

表 5-19　类 SQLiteReadActivity 中方法 onCreate 的代码

序号	代　　码
01	import android.app.Activity;
02	import android.os.Bundle;
03	import android.widget.TextView;
04	import com.example.app0504.util.DBHelper;
05	import com.example.app0504.util.User;
06	public class SQLiteReadActivity extends Activity {
07	@Override
08	protected void onCreate(Bundle savedInstanceState) {
09	super.onCreate(savedInstanceState);　// 调用父类方法
10	setContentView(R.layout.result);　// 设置布局文件
11	TextView username = (TextView) findViewById(R.id.tvUsername2);
12	TextView password = (TextView) findViewById(R.id.tvPassword2);
13	DBHelper helper = new DBHelper(SQLiteReadActivity.this);
14	User user = helper.query(1);
15	username.setText("用户名：" + user.getUsername());　// 显示用户名
16	password.setText("密　　码：" + user.getPassword());　// 显示密码
17	}
18	}

9．在 AndroidManifest.xml 文件中注册 2 个 Activity 并配置启动项

类 SQLiteReadActivity 定义完成后，必须在 AndroidManifest.xml 文件编写代码进行注册，否则无法打开，另外还需要配置启动项，代码如表 5-20 中第 17 行所示。

表 5-20　AndroidManifest.xml 文件的代码

序号	代　　码
01	<?xml version="1.0" encoding="utf-8"?>
02	<manifest xmlns:android="http://schemas.android.com/apk/res/android"
03	package="com.example.app0504" >
04	<application
05	android:allowBackup="true"
06	android:icon="@mipmap/ic_launcher"
07	android:label="@string/app_name"
08	android:theme="@style/AppTheme" >
09	<activity
10	android:name=".SQLiteWriteActivity"
11	android:label="@string/app_name" >
12	<intent-filter>
13	<action android:name="android.intent.action.MAIN" />
14	<category android:name="android.intent.category.LAUNCHER" />
15	</intent-filter>
16	</activity>
17	<activity android:name=".SQLiteReadActivity"/>
18	</application>
19	</manifest>

10．程序运行与功能测试

Android 项目 App0504 运行成功时，登录一个登录界面，在“用户名”文本框中输入正确的用户名，这里输入“admin”，在“密码”文本框中输入对应的密码，这里输入“123456”，如图 5-6 所示，然后单击【登录】按钮，弹出另一个界面显示用户登录时输入的用户名和密码，如图 5-7 所示。



图 5-6　在用户登录界面输入正确的用户名和密码



图 5-7　显示用户登录输入的用户名和密码

【任务 5-5】预览选择的系统图片

【任务描述】

单击应用程序界面的按钮，打开系统的图片浏览工具进行图片选择，然后在界面中预览所选图片，试编程实现这一功能。

【知识索引】

（1）Activity 类及 getContentResolver()、onActivityResult()方法。

（2）ContentProvider 类、Uri 类。

（3）Intent 类及 startActivityForResult()、getData()方法。

（4）Cursor 类及 moveToFirst()、getColumnIndex()、getString()、close()等。

（5）BitmapFactory 类及 decodeFile()方法。

（6）ContentResolver 对象及 query()方法。

（7）LinearLayout 布局、ImageView 控件。

【实施过程】

1．创建 Android 项目 App0505 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0505，将本任务所需的图片导入或复制到 res/drawable 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 1 个 ImageView 控件和 1 个 Button 控件，设置 ImageView 控件的 id 属性为 imgView，将 Button 控件的 id 属性设置为 btnSelectPicture、text 属性设置为“选择图片”。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明 ImageView 对象和 Button 对象，代码如下所示。

```
private Button btnImage;
private ImageView loadImage;
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中得到 ImageView 对象和 Button 对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 5-21 所示。

表 5-21　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 绑定控件
06	btnImage = (Button) findViewById(R.id.btnSelectPicture);
07	loadImage = (ImageView) findViewById(R.id.imgView);
08	// 绑定控件事件
09	setListener();
10	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()实现代码如表 5-22 所示，当用户单击按钮时定义 Intent 对象，设置其属性为 ACTION_PICK，可以从系统中选择图片，然后通过调用方法 startActivityForResult()启动系统的图片浏览程序进行图片选择。

表 5-22　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	// 设置事件
03	btnImage.setOnClickListener(new View.OnClickListener() {
04	@Override
05	public void onClick(View arg0) {
06	Intent intent = new Intent(Intent.ACTION_PICK,
07	android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
08	//启动 intent
09	startActivityForResult(intent, 1);
10	}
11	});
12	}

当图片选择完毕后，系统自动回调 onActivityResult()方法，其代码如表 5-23 所示。其中 data 得到回调的 intent 参数中包含用户所选择的图片信息，通过 ContentResolver 对象得到此回调信息中的图片，然后设置给 ImageView 对象就可以预览所选择图片。

表 5-23　回调方法 onActivityResult()的代码

序号	代　　码
01	@Override
02	protected void onActivityResult(int requestCode, int resultCode, Intent data) {
03	super.onActivityResult(requestCode, resultCode, data);
04	if (requestCode == 1 && resultCode == RESULT_OK　&& null != data) {
05	// 获取数据
06	Uri selectedImage = data.getData();
07	String[] filePathColumn = { MediaStore.Images.Media.DATA };
08	// 查询游标
09	Cursor cursor = getContentResolver().query(selectedImage,filePathColumn, null, null, null);
10	cursor.moveToFirst();
11	// 获取数据
12	int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
13	String picturePath = cursor.getString(columnIndex);
14	cursor.close();
15	// 设置图片
16	loadImage.setImageBitmap(BitmapFactory.decodeFile(picturePath));
17	}
18	}

4．程序运行与功能测试

Android 项目 App0505 的初始运行状态如图 5-8 所示，单击【选择图片】按钮，弹出系统图片列表，选择 1 张图片后即可返回图片浏览界面预览图片。



图 5-8　Android 项目 App0505 的初始运行状态

【任务 5-6】实现添加与查询联系人

【任务描述】

试编程实现在 Android 应用程序界面添加与查询联系人的功能。

【知识索引】

（1）Intent 类及 setType()、putExtra()、startActivity()等方法。

（2）ContentResolver 类及 query()方法。

（3）Activity 类及 getContentResolver()方法。

（4）StringBuilder 类及 append()方法。

（5）Cursor 类及 moveToNext()、getColumnIndex()、getInt()、getString()、close()等方法。

【实施过程】

1．创建 Android 项目 App0506

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0506。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 2 个 EditText 控件、2 个 Button 控件和 1 个 TextView 控件，将 EditText 控件的 id 属性分别设置为 etName 和 etPhone、hint 属性分别设置为“联系人姓名”和“联系人电话”，将 Button 控件的 id 属性分别设置为 btnAdd 和 btnFind、text 属性分别设置为“添加联系人”和“查询联系人”，将 TextView 控件的 id 属性设置为 tvResult、layout_gravity 属性设置为“center_horizontal”。

3．完善类 MainActivity 与实现程序功能

（1）声明对象与数组

在 MainActivity 类定义中，首先声明多个对象和数组，代码如下所示：

```
private Button btn1,btn2;
private EditText name;
private EditText phone;
private TextView result ;
private String[] columns = { ContactsContract.Contacts._ID,   // 获得 ID 值
                ContactsContract.Contacts.DISPLAY_NAME,       // 获得姓名
                Phone.NUMBER,                                 // 获得电话
                Phone.CONTACT_ID, };
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中得到 EditText 对象、Button 对象和 TextView 对象，然后通过自定义

方法 setListener()设置控件对象的监听器，其代码如表 5-24 所示。

表 5-24　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	btn1 = (Button) findViewById(R.id.btnAdd);
06	btn2 = (Button) findViewById(R.id.btnFind);
07	name = (EditText) findViewById(R.id.etName);
08	phone = (EditText) findViewById(R.id.etPhone);
09	result = (TextView) findViewById(R.id.tvResult);
10	//设置对象的监听器
11	setListener();
12	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()实现代码如表 5-25 所示，当用户单击【添加联系人】按钮时设置 intent 的 action 为 Contacts.Intents.Insert.ACTION，实现添加联系人的姓名和电话。当用户单击【查询联系人】按钮时调用自定义方法 getQueryData()获取联系人的信息，然后在 TextView 控件中显示查询结果。

表 5-25　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	// 设置 btn1 的单击监听器
03	btn1.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View v) {
06	Intent intent = new Intent(ContactsContract.Intents.Insert.ACTION);
07	intent.setType(ContactsContract.Contacts.CONTENT_TYPE);
08	//添加联系人的姓名
09	intent.putExtra(ContactsContract.Intents.Insert.NAME, name.getText().toString());
10	//添加联系人的电话
11	intent.putExtra(ContactsContract.Intents.Insert.PHONE, phone.getText().toString());
12	//启动 activity
13	startActivity(intent);
14	}
15	});
16	// 设置 btn2 的单击监听器
17	btn2.setOnClickListener(new OnClickListener() {
18	@Override

续表

序号	代　　码
19	public void onClick(View v) {
20	result.setText(getQueryData());
21	}
22	});
23	}

自定义方法 getQueryData()的代码如表 5-26 所示。

表 5-26　自定义方法 getQueryData()的代码

序号	代　　码
01	private String getQueryData() {
02	StringBuilder sb = new StringBuilder();　　// 用于保存字符串
03	ContentResolver resolver = getContentResolver();　// 获得 ContentResolver 对象
04	Cursor cursor = resolver.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
05	sb.append("序号　姓名　联系电话\n");
06	while (cursor.moveToNext()) {
07	int idIndex = cursor.getColumnIndex(columns[0]);　　// 获得 ID 值的索引
08	int displayNameIndex = cursor.getColumnIndex(columns[1]);　// 获得姓名索引
09	int id = cursor.getInt(idIndex);　　// 获得 ID
10	String displayName = cursor.getString(displayNameIndex);　　// 获得名称
11	Cursor phone = resolver.query(Phone.CONTENT_URI, null, columns[3]
12	+ "=" + id, null, null);
13	while (phone.moveToNext()) {
14	int phoneNumberIndex = phone.getColumnIndex(columns[2]);　　// 获得电话索引
15	String phoneNumber = phone.getString(phoneNumberIndex);　　// 获得电话
16	sb.append("　"+id+".　"+displayName + "　" + phoneNumber + "\n");　//保存数据
17	}
18	}
19	cursor.close();　　// 关闭 Cursor
20	return sb.toString();　　// 返回查询结果
21	}

4．在 AndroidManifest.xml 文件中增加读取联系人记录的权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加读取联系人记录的权限。

```
<uses-sdk android:minSdkVersion="17" />
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

5．程序运行与功能测试

Android 项目 App0506 的初始运行状态如图 5-9 所示。

在“联系人姓名”文本框中输入姓名，这里输入“xiatian”，在“联系人电话”文本框中输入电话号码，这里输入“18837216666”，如图 5-10 所示。



图 5-9　Android 项目 App0506 的初始运行状态



图 5-10　在文本框中输入联系人的姓名和电话

然后单击【添加联系人】按钮，弹出【新增联系人】界面，且显示前一步输入的姓名和电话，如图 5-11 所示，在该界面中还可以添加工作单位、电子邮件和地址等信息，添加完成后返回前一界面，单击【查询联系人】按钮，显示所有联系人信息列表，如图 5-12 所示。



图 5-11　【新增联系人】界面



图 5-12　联系人信息列表



单元小结

本单元主要介绍了 Android 系统提供的四种数据存储方式：SharePreference、SQLite、Content Provider 和 File，通过完成实例任务，学会了使用 SQLite 数据库存储数据、使用 SharedPreference 对象存储数据、使用 SharedPreference 对象存储数据和使用 File 对象存储数据。



单元习题

一、填空题

1．Android 中的数据存储方式主要有 5 种，分别是（　　）、（　　）、（　　）、（　　）和网络。

2．Android 中的文件可以存储在（　　）和（　　）中。

3．SharedPreferences 是一个轻量级的存储类，主要用于存储一些应用程序的（　　）。

4．创建数据库可以通过使用（　　）方法创建数据库。另外还可以通过写一个继承（　　）类的方式创建数据库。

5．要查询 SQLite 数据库中的信息需要使用（　　）接口，使用完毕后调用（　　）关闭。

6．ContentProvider 提供了对数据进行增、删、改、查的方法，分别为（　　）、（　　）、（　　）和（　　）。

7．ContentProvider 用于（　　）和（　　）数据，是 Android 中不同应用程序之间共享数据的接口。

8．在 Android 应用程序中，使用 ContentProvider 暴露自己数据，通过（　　）对暴露的数据进行操作。

9．SharedPreferences 本质上是一个 XML 文件，所存储的数据是以（　　）的格式保存在 XML 文件中。

10．当用户将文件保存至 SD 卡时，需要在清单文件中添加权限（　　）。

11．ContentResolver 可以通过 ContentProvider 提供的（　　）进行数据操作。

12．ContentProvider 与 Activity 一样，创建时首先会调用（　　）方法。

二、选择题

1．下列文件操作权限中，指定文件内容可以追加的是（　　）。

A．MODE_PRIVATE　　B．MDOE_WORLD_READABLE

C．MODE_APPEND　　D．MODE_WORLD_WRITEABLE

2．下列代码中，用于获取 SD 卡路径的是（　　）。

A．Environment.getSD();　　B．Environment.getExternalStorageState();

C．Environment.getDSDirectory();　　D．Environment.getExternamStorageDirectory();

3．下列选项中，关于文件存储数据的说法错误的是（　　）。

A．文件存储是以流的形式来操作数据的

B．文件存储可以将数据存储到 SD 卡中

C．文件存储可以将数据存储到内存中

D．Android 程序中只能使用文件存储数据

4．如果要将程序中的私有数据分享给其他应用程序，可以使用的是（　　）。

A．文件存储　　B．SharedPriferences　　C．ContentProvider　　D．SQLite

5．下列命令中，属于 SQLite 下的命令是（　　）。

A．shell　　B．push　　C．quit　　D．keytool

6．以下哪个方法能够实现数据库的数据插入？（　　）

A．onCreate　　B．onUpgrade　　C．execSQL　　D．rawQuery

7．以下哪种数据库操作不能使用 execSQL 方法执行？（　　）

插入记录　　B．删除记录　　C．查询记录　　D．创建数据表

8．使用 SQLite 数据库进行查询操作后，必须要做的操作是（　　）。（多选题）

A．关闭数据　　B．直接退出

C．关闭 Cursor　　D．使用 quit 函数退出

三、简答题

1．简述几种 Android 数据存储的方法和特点。

2．简要说明 SQLite 数据库创建的过程。

3．简述使用 Sharedpreference 如何存储数据。

4．简要说明 ContentProvider 对外共享数据的好处。

单元 6

Android 的服务与广播应用程序设计

Service(服务)是 Android 系统中的四大组件之一(Activity、Service、BroadcastReceiver、ContentProvider)，是一种在后台执行长时间程序运行、无需与用户交互的组件，并且可以和其他组件进行交互。

BroadcastReceiver（广播接收者），顾名思义，它是用来接收来自系统和应用中的广播。Broadcast 是 Android 中一种广泛运用的在应用程序之间传输信息的机制，它用于发送广播，把要发送的信息和用于过滤的信息(如 Action、Category)装入一个 Intent 对象，然后通过调用 Context.sendBroadcast()、sendOrderBroadcast()或 sendStickyBroadcast()方法，把 Intent 对象以广播方式发送出去。而 BroadcastReceiver 是对发送出来的 Broadcast 进行过滤接受并响应的一类组件，它用于接收广播，当 Intent 发送以后，所有已经注册的 BroadcastReceiver 会检查注册时的 IntentFilter 是否与发送的 Intent 相匹配，若匹配则就会调用 BroadcastReceiver 的 onReceive()方法。

教学导航

教学目标	（1）理解与掌握 Service 的基本概念、创建与注册、启动流程、生命周期、类型等内容 （2）理解与掌握 BroadcastReceiver 的基本概念、生命周期、注册方式、类型等内容 （3）学会 Service 的典型应用 （4）学会 BroadcastReceiver 的典型应用
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	8 课时

知识导读

6.1 Service（服务）

1．Service 的基本概念

Service(服务)是一个没有用户界面、在后台运行的应用组件，相当于后台运行的 Activity。其他应用组件能够启动 Service，并且当用户切换到另外的应用场景，Service 将持续在后台运行。另外，一个组件能够绑定到一个 Service 与之交互（IPC 机制），例如，一个 Service 可能会处理网络操作、播放音乐、操作文件 I/O 或者与内容提供者（Content Provider）交互，所有这些活动都是在后台进行的。Service 和其他组件一样，都是运行在主线程中，因此不能用它

来做耗时的请求或者动作。

本质上，一个服务可以采取两种状态：

（1）Started 状态

一个应用程序组件（例如活动）通过调用 StartService()启动一个服务开始，以后服务可以无限期地在后台运行，即使启动它的组件被破坏。

（2）Bound 状态

一个应用程序组件可以调用 bindService()方法绑定服务，绑定服务提供客户端与服务器间的接口，允许组件之间进行交互、发送请求、得到结果，这样可以实现跨进程间通信。

2．Service 的回调方法

要创建一个服务，需要创建一个 Java 类，扩展 Service 基类或者它的子类。Service 基类定义了各种回调方法，如表 6-1 所示。但是也并不需要实现所有的回调方法，重要的是要了解每一个回调方法以及实现，以确保应用程序能如用户所期望的行为方式运行。

表 6-1　Service 的回调方法

回调方法名称	使 用 说 明
onStartCommand()	当另一个组件（例如 Activity）调用 startService()方法请求服务启动时，调用该方法。如果开发人员实现该方法，则需要在任务完成时调用 stopSelf()或 stopService()方法停止服务（如果仅想提供绑定，则不必实现该方法）
onBind()	当其他组件通过调用 bindService()绑定服务时调用这个方法。在该方法的实现中，必须提供客户端与服务通信的接口。该方法必须实现，但是如果不想允许绑定，则返回 null
onCreate()	当服务第一次创建时,系统调用该方法执行一次性建立过程（在系统调用 onStartComman()或 onBind()方法前）。如果服务已经运行，该方法不被调用
onDestroy()	当服务不再使用并将被销毁时，系统调用这个方法。服务应该实现该方法用于清理，如线程注册的侦听器、接收器等资源

3．Service 的创建与注册

创建一个名为 Android 项目,在该项目中创建一个继承自 Service 类的子类 LocalService，该类有一个抽象方法 onBind()，必须在子类中实现。

服务 Service 需要在 AndroidManifest.xml 文件中进行注册，注册类 LocalService 的代码如下所示。

```
<service android:name=".LocalService" />
```

该代码应位于 application 节点内，与 activity 组件的注册位于同一层次。

4．Service 的启动流程

Service 有“启动”和“绑定”两种状态。通过 startService()启动的服务处于“启动的”状态，一旦启动，Service 就在后台运行，即使启动它的应用组件已经被销毁了。通常 started 状态的 Service 执行单任务并且不返回任何结果给启动者。例如当下载或上传一个文件，当这项操作完成时，Service 应该停止它本身。还有一种“绑定”状态的 Service，通过调用 bindService()来启动，一个绑定的 Service 提供一个允许组件与 Service 交互的接口，可以发送请求、获取返回结果，还可以通过跨进程通信来交互（IPC）。绑定的 Service 只有当应用组件绑定后才能运行，多个组件可以绑定一个 Service，当调用 unbind()方法时，这个 Service 就会被销毁了。

Service 的启动有两种方式：context.startService()和 context.bindService()。

（1）context.startService()方式启动服务

下面通过一个简单的实例来说明使用 startService()方式启动服务。

首先创建一个 Android 项目，在该项目中创建一个继承自 Service 类的子类 LocalService，子类 LocalService 的代码如表 6-2 所示，分别重写了 onCreate()、onStartCommand()和 onDestroy()方法，通过观察 logcat 窗口中输出的日志信息可以了解服务执行的整个过程。

表 6-2　子类 LocalService 的代码

序号	代　　码
01	package com.example.exampleapp0601;
02	import android.app.Service;
03	import android.content.Intent;
04	import android.os.IBinder;
05	import android.util.Log;
06	public class LocalService extends Service {
07	private static final String TAG = "LocalService";
08	@Override
09	public IBinder onBind(Intent intent) {
10	return null;
11	}
12	@Override
13	public void onCreate() {
14	Log.i(TAG, "onCreate");
15	super.onCreate();
16	}
17	@Override
18	public int onStartCommand(Intent intent, int flags, int startId) {
19	Log.i(TAG, "onStartCommand");
20	return START_STICKY;
21	}
22	@Override
23	public void onDestroy() {
24	Log.i(TAG, "onDestroy");
25	super.onDestroy();
26	}
27	}

在界面中添加两个 Button 控件，分别用于开启服务和关闭服务，XML 文件对应的代码如表 6-3 所示。

表 6-3　XML 文件对应的代码

序号	代　　码
01	<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	xmlns:tools="http://schemas.android.com/tools"

续表

序号	代　　码
03	……
04	android:orientation="horizontal">
05	<Button
06	android:id="@+id/btnStart"
07	android:layout_width="wrap_content"
08	android:layout_height="wrap_content"
09	android:onClick="start"
10	android:text="开启服务"　/>
11	<Button
12	android:id="@+id/btnStop"
13	android:layout_width="wrap_content"
14	android:layout_height="wrap_content"
15	android:onClick="stop"
16	android:text="关闭服务"　/>
17	</LinearLayout>

使用 startService()方式启动服务的界面外观如图 6-1 所示。



图 6-1　使用 startService()方式启动服务的界面外观

在 MainActivity.java 文件中编写代码实现所需功能，对应的代码如表 6-4 所示，实现了【开启服务】与【关闭服务】按钮的单击事件。

表 6-4　MainActivity.java 文件的代码

序号	代　　码
01	package com.example.exampleapp0601;
02	import android.view.View;
03	import android.app.Activity;
04	import android.os.Bundle;
05	import android.content.Intent;
06	public class MainActivity extends Activity {
07	@Override
08	protected void onCreate(Bundle savedInstanceState) {
09	super.onCreate(savedInstanceState);
10	setContentView(R.layout.activity_main);
11	}
12	// 开启服务
13	public void start(View view) {
14	Intent intent=new Intent(this, LocalService.class);
15	startActivity(intent);

续表

序号	代　码
16	} // 停止服务 public void stop(View view){ Intent intent=new Intent(this, LocalService.class); stopService(intent); }
17	
18	
19	
20	
21	
22	}

当前程序运行时，单击界面上的【开启服务】按钮，从 logcat 窗口中输出服务创建的 Log 信息，从日志信息可以看出，服务创建时首先执行的是 onCreate()方法，当服务启动时执行的是 onStartCommand()。需要注意的是，onCreate()方法只有在服务创建时执行，而 onStartCommand()方法则是每次启动服务时调用。

单击【关闭服务】按钮时，在 logcat 窗口输出服务销毁的 Log 信息，从日志信息可以看出，当单击【关闭服务】按钮时，服务执行 onDestroy()方法。

startService()的启动流程如图 6-3 所示，其过程描述如下：

```
    context.startService()->  onCreate()->  onStart()->  Service  running->
context.stopService()-> onDestroy()-> Service stop
```

如果 Service 还没有运行，则 Android 先调用 onCreate()，然后调用 onStart()；如果 Service 已经运行，则只调用 onStart()，所以一个 Service 的 onStart 方法可能会重复调用多次。

如果直接退出而没有调用 stopService 的话，Service 会一直在后台运行，该 Service 的调用者再启动起来后可以通过 stopService 关闭 Service。

所以调用 startService 的生命周期为：onCreate→onStart (可多次调用)→onDestroy。

（2）context.bindService()方式启动服务

当程序使用 startService()和 stopService()启动、关闭服务时，服务与调用者之间基本不存在太多的关联，也无法与访问者进行通信、数据交互等。如果服务需要与调用者进行访问调用和数据交互时，应该使用 bindService()和 unbindService()启动、关闭服务。

下面通过一个简单的实例来说明使用 bindService()方式启动服务。

首先，创建一个 Android 项目，在该项目中创建一个继承自 Service 类的子类 MyBindService，子类 MyBindService 的代码如表 6-5 所示，分别重写了 onCreate()、onBind()和 onUnbind()方法，通过观察 logcat 窗口中输出的日志信息可以了解服务执行的整个过程。

表 6-5　子类 MyBindService 的代码

序号	代　码
01	package com.example.exampleapp0602;
02	import android.app.Service;
03	import android.content.Intent;
04	import android.os.Binder;
05	import android.os.IBinder;
06	import android.util.Log;

续表

序号	代　　码
07	public class MyBindService extends Service {
08	@Override
09	public void onCreate(){
10	super.onCreate();
11	Log.i("MyBinService","创建服务，调用 onCreate()");
12	}
13	@Override
14	public IBinder onBind(Intent intent) {
15	Log.i("MyBinService","绑定服务，调用 onBind()");
16	return new MyBinder();
17	}
18	@Override
19	public boolean onUnbind(Intent intent){
20	Log.i("MyBinService","解绑服务，调用 onUnbind()");
21	return super.onUnbind(intent);
22	}
23	}

在界面中添加两个 Button 控件，分别用绑定服务与解绑服务，XML 文件对应的代码如表 6-6 所示。

表 6-6　XML 文件对应的代码

序号	代　　码
01	<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	xmlns:tools="http://schemas.android.com/tools"
03	……
04	android:orientation="horizontal">
05	<Button
06	android:id="@+id/btnBind"
07	android:layout_width="wrap_content"
08	android:layout_height="wrap_content"
09	android:onClick="start"
10	android:text="绑定服务"　/>
11	<Button
12	android:id="@+id/btnUnbind"
13	android:layout_width="wrap_content"
14	android:layout_height="wrap_content"
15	android:onClick="stop"
16	android:text="解绑服务"　/>
17	</LinearLayout>

使用 bindService()方式启动服务的界面外观如图 6-2 所示。



图 6-2　使用 bindService()方式启动服务的界面外观

MainActivity.java 文件中的代码如表 6-7 所示，实现了【绑定服务】与【解绑服务】按钮的单击事件。

表 6-7　MainActivity.java 文件的代码

序号	代　　码
01	package com.example.exampleapp0602;
02	import android.app.Activity;
03	import android.content.ComponentName;
04	import android.content.ServiceConnection;
05	import android.os.Bundle;
06	import android.os.IBinder;
07	import android.view.View;
08	import android.util.Log;
09	import android.content.Intent;
10	//创建 MyConn 类，用于实现连接服务
11	class MyConn implements ServiceConnection {
12	public void onServiceConnected(ComponentName name,IBinder service){
13	Log.i("MainActivity","服务成功绑定");
14	}
15	//当服务失去连接时调用的方法
16	public void onServiceDisconnected(ComponentName name){
17	}
18	}
19	public class MainActivity extends Activity {
20	private MyConn conn;
21	@Override
22	protected void onCreate(Bundle savedInstanceState) {
23	super.onCreate(savedInstanceState);
24	setContentView(R.layout.activity_main);
25	}
26	//绑定服务
27	public void btnBind(View view){
28	if(conn==null){
29	conn=new MyConn();
30	}
31	Intent intent=new Intent(this,MyBindService.class);
32	//参数 1 是 Intent 对象，参数 2 是连接对象，参数 3 表示如果服务不存在则创建服务
33	bindService(intent,conn,BIND_AUTO_CREATE);
34	}
35	//解绑服务
36	public void btnUnbind(View view){

续表

序号	代　码
37	if(conn==null){
38	unbindService(conn);
39	conn=null;
40	}
41	}
42	}

表 6-7 中的代码第 33 行调用 bindService()方法绑定服务，第 38 行调用 unbindService()方法解绑服务。

context.bindService()的启动流程如图 6-4 所示，其过程描述如下：

```
context.bindService()-> onCreate()-> onBind()-> Service running->
onUnbind()-> onDestroy()-> Service stop
```

onBind()将返回给客户端一个 IBind 接口实例，IBind 允许客户端回调服务的方法，例如得到 Service 的实例、运行状态或其他操作。这时调用者（例如 Activity）会和 Service 绑定在一起，Context 退出了，Service 就会调用 onUnbind→onDestroy 相应退出。

所以调用 bindService 的生命周期为：onCreate→onBind（只一次，不可多次绑定）→onUnbind→onDestory。

在 Service 每一次的开启关闭过程中，只有 onStart 可被多次调用（通过多次 startService 调用），其他 onCreate，onBind，onUnbind，onDestory 在一个生命周期中只能被调用一次。

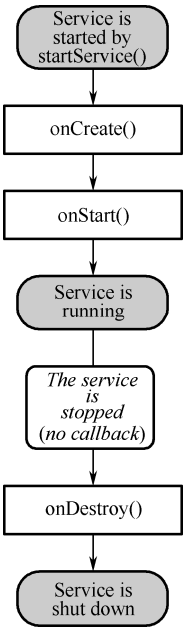


图 6-3　context.startService()的启动流程

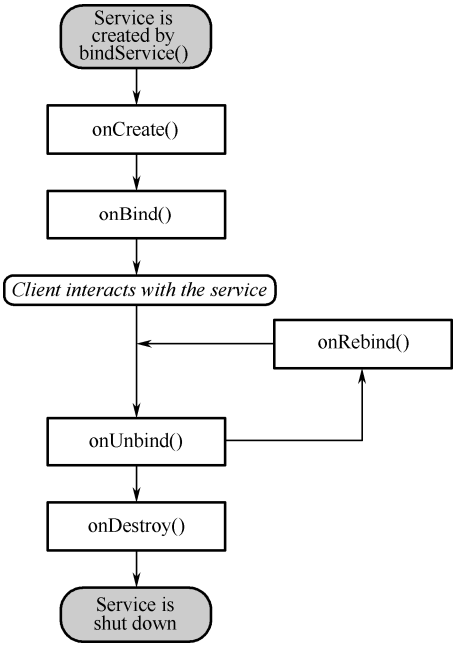


图 6-4　context.bindService()的启动流程

如果只是想要启动一个程序，在后台服务长期进行某项任务，那么使用 startService 便可以了。如果想要与正在运行的 Service 取得联系，有两种方法，一种是使用 broadcast，另一种是使用 bindService，前者的缺点是如果交流较为频繁，容易造成性能上的问题，并且 BroadcastReceiver 本身执行代码的时间是很短的，而后者则没有这些问题，因此我们选择使用 bindService，此时也同时在使用 startServic 和 bindService 了，这在 Activity 中更新 Service 的某些运行状态是有用的。另外如果服务只是公开一个远程接口，供连接上的客户端远程调用执行方法，此时可以不让服务一开始就运行，而只用 bindService，这样在第一次 bindService 的时候才会创建服务的实例运行它，这会节约很多系统资源。

5．Service 的生命周期

Service 的生命周期并不像 Activity 那么复杂，它只继承了 onCreate()、onStart()、onDestroy()三个方法。当我们第一次启动 Service 时，先后调用了 onCreate()、onStart()这两个方法；当停止 Service 时，则执行 onDestroy()方法。

这里需要注意的是，如果 Service 已经启动了，当我们再次启动 Service 时，不会在执行 onCreate()方法，而是直接执行 onStart()方法。它可以通过 Service.stopSelf()方法或者 Service.stopSelfResult()方法来停止自己，只要调用一次 stopService()方法便可以停止服务，无论调用了多少次的启动服务方法。

（1）被启动服务的生命周期

如果一个 Service 被某个 Activity 调用 Context.startService 方法启动，那么不管是否有 Activity 使用 bindService 绑定或 unbindService 解除绑定到该 Service，该 Service 都在后台运行。如果一个 Service 被 startService 方法多次启动，那么 onCreate 方法只会调用一次，onStart 将会被调用多次（对应调用 startService 的次数），并且系统只会创建 Service 的一个实例。该 Service 将会一直在后台运行，而不管对应程序的 Activity 是否在运行，直到被调用 stopService，或自身的 stopSelf 方法。当然如果系统资源不足，Android 系统也可能结束服务。

（2）被绑定服务的生命周期

如果一个 Service 被某个 Activity 调用 Context.bindService 方法绑定启动，不管调用 bindService 几次，onCreate 方法都只会调用一次，同时 onStart 方法始终不会被调用。当连接建立之后，Service 将会一直运行，除非调用 Context.unbindService 断开连接或者之前调用 bindService 的 Context 不存在了（如 Activity 被 finish 的时候），系统将会自动停止 Service，对应 onDestroy 将被调用。

（3）被启动又被绑定的服务的生命周期

如果一个 Service 又被启动又被绑定，则该 Service 将会一直在后台运行。并且不管如何调用，onCreate 始终只会调用一次，对应 startService 调用多少次，Service 的 onStart 便会调用多少次。调用 unbindService 将不会停止 Service，而必须调用 stopService 或 Service 的 stopSelf 来停止服务。

（4）当服务被停止时清除服务

当一个 Service 被终止（可能是调用 stopService 或者调用 stopSelf 或者不再有绑定的连接）时，onDestroy 方法将会被调用，在这里可以做一些清除工作，例如停止在 Service 中创建并运行的线程。

【注意事项】:

① 调用 bindService 绑定到 Service 时，应保证在某处调用 unbindService 解除绑定。

② 使用 startService 启动服务之后，不管是否使用 bindService，都要使用 stopService 停止服务。

③ 同时使用 startService 与 bindService 时，Service 的终止，需要同时调用 unbindService 与 stopService，才能终止 Service，不管 startService 与 bindService 的调用顺序如何。如果先调用 unbindService，此时服务不会自动终止，再调用 stopService 之后服务才会停止；如果先调用 stopService，此时服务也不会终止，而再调用 unbindService 或者之前调用 bindService 的 Context 不存在了。

④ 在 SDK2.0 及其以后的版本中，对应的 onStart 方法已经变为 onStartCommand，不过之前的 onStart 仍然有效。这意味着，如果开发的应用程序用的 SDK 2.0 及其以后的版本，那么应当使用 onStartCommand 而不是 onStart。

6．Service 的类型

Service（服务）一般分为两种：

（1）本地服务（Local Service）

Local Service 用于应用程序内部。在 Service 可以调用 Context.startService()启动，调用 Context.stopService()结束。在内部可以调用 Service.stopSelf()或 Service.stopSelfResult()来自己停止。无论调用了多少次 startService()，都只需调用一次 stopService()来停止。

（2）远程服务（Remote Service）

Remote Service 用于 Android 应用程序之间。可以定义接口并把接口暴露出来，以便其他应用进行操作。客户端建立到服务对象的连接，并通过那个连接来调用服务。调用 Context.bindService()方法建立连接，并启动，以调用 Context.unbindService()关闭连接。多个客户端可以绑定至同一个服务。如果服务此时还没有加载，bindService()会先加载它。提供给可被其他应用程序复用，例如定义一个天气预报服务，提供与其他应用调用即可。

7．在 AndroidManifest.xml 文件中配置 Service 元素的常见选项

在 AndroidManifest.xml 文件中配置 Service 元素的示例代码如下：

```
<service
    android:name=".MyBindService"
    android:enabled="true"
    android:exported="true" >
</service>
```

在 AndroidManifest.xml 文件中配置 Service 元素的常见选项如下：

① android:name：实现服务的 Service 子类名称，类名应该是一个包含包名的完整名称，例如“com.example.exampleapp0601.LocalService”。为了简便，如果名称的第一个符号是点号，例如“.LocalService”，则会增加在 AndroidManifest.xml 文件中定义包名，例如“com.example.exampleapp0601”。

② android:label：显示给用户的服务名称，如果此项不设置，那么默认显示的服务名则为类名。

③ android:icon：表示服务图标，该属性必须设置成包含图片定义的可绘制资源引用。如果没有设置，使用应用程序的图标替代。

④ android:permission：声明此服务的权限，这意味着只有提供了该权限的应用才能控制或连接此服务。如果 startService()、bindService()或 stopService()方法调用者没有被授权，则该调用无效，并且 Intent 对象也不会发送给服务。如果该属性没有设置，使用<application>标签的 permission 属性设置给服务。如果<application>和<service>标签的 permission 属性都未设置，则服务不受权限保护。

⑤ android:process：表示运行的进程名称。通常，应用程序的全部组件运行于为应用程序创建的默认进程，它与应用程序包名相同。如果设置了此项，那么将会在包名后面加上这段字符串表示该服务是否运行在另外一个进程。

⑥ android:enabled：如果此项设置为 true，那么 Service 将会默认被系统启动。

⑦ android:exported：表示该服务是否能够被其他组件调用，true 表示可以，false 表示不可以，其默认值为 false。当其值设置为 false 时，只有同一个应用程序的组件或者具有相同用户 id 的应用程序才能启动或者绑定到服务。

6.2 BroadcastReceiver（广播）

在 Android 系统中，广播体现在方方面面，例如当开机完成后系统会产生一条广播，接收到这条广播就能实现开机启动服务的功能；当网络状态改变时系统会产生一条广播，接收到这条广播就能及时地做出提示和保存数据等操作；当电池电量改变时，系统会产生一条广播，接收到这条广播就能在电量低时告知用户及时保存进度。

Android 中的广播机制非常出色，很多事情原本需要开发者亲自操作的，现在只需等待广播告知自己就可以了，大大减少了开发的工作量和开发周期。

1．BroadcastReceiver 的基本概念

BroadcastReceiver 用于异步接收广播，广播的发送是通过调用 Context.sendBroadcast()、Context.sendOrderedBroadcast()、Context.sendStickyBroadcast()来实现的。通常一个广播可以被订阅了此 Intent 的多个广播接收者所接收，广播接收者和 JMS 中的 Topic 消息接收者很相似。

广播接收器通过调用 BroadcastReceiver()方法接收广播，对广播的通知做出反应，很多广播都产生于系统代码，例如时区改变的通知、电池电量不足、用户改变了语言偏好，或者开机启动等。广播接收器没有用户界面，但是它可以为它们接收到信息启动一个 Activity 或者使用 NotificationManager 来通知用户。

BroadcastReceiver（广播接收者）是一个系统全局的监听器，用于监听系统全局的 Broadcast 消息，所以它可以很方便地进行系统组件之间的通信。BroadcastReceiver 虽然是一个监听器，但是它和之前用到的 OnXxxListener 监听器不同，那些只是程序级别的监听器，运行在指定程序的所在进程中，当程序退出的时候，OnXxxListener 监听器也就随之关闭了，但是 BroadcastReceiver 属于系统级的监听器，它拥有自己的进程，只要存在与之匹配的 Broadcast 被以 Intent 的形式发送出来，BroadcastReceiver 就会被激活。

2．BroadcastReceiver 的生命周期

虽然 BroadcastReceiver 同属 Android 的四大组件，但它也有自己独立的生命周期，但是和 Activity、Service 又不同。当在系统注册一个 BroadcastReceiver 之后，每次系统以一个 Intent 的形式发布 Broadcast 的时候，系统都会创建与之对应的 BroadcastReceiver 广播接收者实例，并自动触发它的 onReceive()方法，当 onReceive()方法被执行完成之后，BroadcastReceiver 实

例就会被销毁。虽然它独自享用一个单独的进程，但也不是没有限制的，如果BroadcastReceiver.onReceive()方法不能在几秒内执行完成，Android 系统就会认为该BroadcastReceiver 对象无响应，然后弹出 ANR（Application No Response）对话框，所以不要在 BroadcastReceiver.onReceive()方法内执行一些耗时的操作。

BroadcastReceiver 的使用基本流程如下：注册 receriver（静态或动态）→调用 sendBroadcast()方法发送广播→调用 onReceive(Context context, Intent intent)方法处理广播→调用 startService()方法启动服务→调用 stopService()方法关闭服务。

一个 BroadcastReceiver 对象只有在被调用 onReceive(Context, Intent)方法时才有效，当从该方法返回后，该对象就无效了，其生命周期结束。

从这个特征可以看出，在所调用的 onReceive(Context, Intent)方法里，不能有过于耗时的操作，不能使用线程来执行。对于耗时的操作，应该在 startService 中来完成。因为当得到其他异步操作所返回的结果时，BroadcastReceiver 可能已经无效了。

如果需要根据广播内容完成一些耗时的操作，一般考虑通过 Intent 启动一个 Service 来完成该操作，而不应该在 BroadcastReceiver 中开启一个新线程完成耗时的操作，因为BroadcastReceiver 本身的生命周期很短，可能出现的情况是子线程还没有结束，BroadcastReceiver 就已经退出的情况，而如果 BroadcastReceiver 所在的进程结束了，该线程就会被标记为一个空线程，根据 Android 的内存管理策略，在系统内存紧张的时候，会按照优先级，结束优先级低的线程，而空线程是优先级最低的，这样就可能导致 BroadcastReceiver 启动的子线程不能执行完成。

3．BroadcastReceiver 的注册方式

BroadcastReceiver 本质上是一个监听器，所以使用 BroadcastReceiver 的方法也是非常简单，只需要继承 BroadcastReceiver，在其中重写 onReceive(Context context,Intent intent)即可。一旦实现了 BroadcastReceiver，并部署到系统中后，就可以在系统的任何位置，通过sendBroadcast、sendOrderedBroadcast 方法发送 Broadcast 给这个 BroadcastReceiver。

但是仅仅继承 BroadcastReceiver 和实现 onReceive()方法是不够的，同为 Android 系统组件，它也必须在 Android 系统中注册，注册一个 BroadcastReceiver 有两种方式：

（1）静态注册

在清单文件 AndroidManifest.xml 的<application/>节点中使用标签< receiver>进行注册，并在标签内用<intent- filter>标签设置过滤器。在<receiver/>节点中用 android:name 属性中指定注册的 BroadcastReceiver 对象，一般还会通过<Intent-filter/>指定<action/>和<category/>，并在<Intent-filter/>节点中通过 android:priority 属性设置 BroadcastReceiver 的优先级，数值越大优先级越高。

静态注册一个广播地址的示例代码如下所示：

```
<receiver android:name=".MyReceiver">
   <intent-filter>
       <action android:name="android.intent.action.MY_BROADCAST"/>
       <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</receiver>
```

配置了以上信息之后，只要是 android.intent.action.MY_BROADCAST 这个地址的广播，MyReceiver 都能够接收的到。

【注意】：这种方式的注册是常驻型的，也就是说当应用关闭后，如果有广播信息传来，MyReceiver 也会被系统调用而自动运行。

（2）动态注册

动态注册需要在代码中动态的指定广播地址并注册，在代码中先定义并设置好一个 IntentFilter 对象，然后在需要注册的地方调用 Context.registerReceiver()方法，需要取消时就调用 Context.unregisterReceiver()方法。如果用动态方式注册的 BroadcastReceiver 的 Context 对象被销毁时，BroadcastReceiver 也就自动取消注册了。另外，如果在使用 sendBroadcast()方法时指定了接收权限，则只有在 AndroidManifest.xml 文件中用<uses- permission>标签声明了拥有此权限的 BroascastReceiver 才会有可能接收到发送来的 Broadcast。同样，如果在注册 BroadcastReceiver 时指定了可接收的 Broadcast 的权限，则只有在 AndroidManifest.xml 文件中用<uses-permission>标签声明了，拥有此权限的 Context 对象所发送的 Broadcast 才能被这个 BroadcastReceiver 所接收。

动态注册的示例代码如下：

```
MyReceiver receiver = new MyReceiver();
IntentFilter filter = new IntentFilter();
filter.addAction("android.intent.action.MY_BROADCAST");
registerReceiver(receiver, filter);
```

【注意】registerReceiver 是 android.content.ContextWrapper 类中的方法，Activity 和 Service 都继承了 ContextWrapper，所以可以直接调用。在实际应用中，我们在 Activity 或 Service 中注册了一个 BroadcastReceiver，当这个 Activity 或 Service 被销毁时如果没有解除注册，系统会报一个异常，提示我们是否忘记解除注册了。所以，记得在特定的地方执行解除注册操作，解除注册的代码如下所示：

```
@Override
protected void onDestroy() {
    super.onDestroy();
    unregisterReceiver(receiver);
}
```

这种动态注册方式与静态注册相反，不是常驻型的，也就是说广播会跟随程序的生命周期。

我们可以根据以上任意一种方法完成注册，当注册完成之后，这个接收者就可以正常工作了。我们可以用以下方式向其发送一条广播，示例代码如下：

```
public void send(View view) {
    Intent intent = new Intent("android.intent.action.MY_BROADCAST");
    intent.putExtra("msg", "hello receiver.");
    sendBroadcast(intent);
}
```

【注意】：sendBroadcast 也是 android.content.ContextWrapper 类中的方法，它可以将一个指定地址和参数信息的 Intent 对象以广播的形式发送出去。

静态注册方式，由系统来管理 receiver，而且程序里的所有 receiver，可以在 XML 文件中一目了然。动态注册方式的注册代码隐藏在代码中，一般在 activity 的 onStart()里面进行注册，在 onStop()里面进行注销。在退出程序前要记得调用 Context.unregisterReceiver()方法。如果在 Activity.onResume()里面注册了，就必须在 Activity.onPause()注销。

虽然 Android 系统提供了两种方式注册 BroadcastReceiver，但是一般在实际开发中，还是会在清单文件 AndroidManifest.xml 进行注册。

4．BroadcastReceiver 的类型

如果有多个接收者都注册了相同的广播地址，能同时接收到同一条广播吗，相互之间会不会有干扰呢？这就涉及普通广播和有序广播的概念了。

当系统以一个 Intent 的形式发送一个 Broadcast 出去之后，所有与之匹配的 BroadcastReceiver 都会被实例化，但是这里是有区别的，根据 Broadcast 的传播方式区别，在系统中有如下两种 Broadcast：

（1）普通广播（Normal Broadcast）

普通广播对于多个接收者来说是完全异步的，通常每个接收者都无需等待即可以接收到广播，接收者相互之间不会有影响。对于这种广播，接收者无法终止广播，即无法阻止其他接收者的接收动作。当一个 Broadcast 被发出之后，所有与之匹配的 BroadcastReceiver 都同时接收到 Broadcast。这种方式的优点是传递效率比较高，其缺点是一个 BroadcastReceiver 不能影响其他响应这个 Broadcast 的 BroadcastReceiver。

（2）有序广播（Ordered Broadcast）

有序广播是同步执行的，也就是说有序广播的接收器将会按照预先声明的优先级依次接受 Broadcast，它是链式结构，每次只发送到优先级较高的接收者那里，优先级越高越先被执行。因为是顺序执行，所有优先级高的接收器，可以把执行结果传播到优先级低的接收者那里，优先级高的接收者有能力终止这个广播，可以通过 abortBroadcast()方法终止 Broadcast 的传播，一旦 Broadcast 的传播被终止，优先级低于它的接收器就不会再接收到这条 Broadcast 了。

虽然系统存在两种类型的 Broadcast，但是一般系统发送出来的 Broadcast 均是有序广播，所以可以通过优先级的控制，在系统内置的程序响应前，对 Broadcast 提前进行响应。这就是短信拦截器、电话拦截器等软件的工作原理。

根据 Broadcast 传播的方式，Context 提供了不同的方法来发布它们：

（1）sendBroadcast()：发送普通广播。

（2）sendOrderedBroadcast()：发送有序广播。

以上两个方法都有多个重载方法，根据不同的场景使用，最简单的莫过于直接传递一个 Intent 来发送一个广播。

下面通过一个简单的实例，分析 BroadcastReceiver 的声明以及如何向这个 BroadcastReceiver 发送消息。

首先创建一个 Java 文件 BasicBroadcast.java，声明一个类 BroadcastReceiver，代码如表 6-8 所示。

表 6-8　类 BroadcastReceiver 对应的代码

序号	代　　码
01	package com.example.exampleapp0603;
02	import android.content.BroadcastReceiver;
03	import android.content.Context;
04	import android.content.Intent;
05	import android.widget.Toast;
06	public class BasicBroadcast extends BroadcastReceiver {
07	@Override
08	public void onReceive(Context context, Intent intent) {
09	Toast.makeText(context,"接收到 Broadcast，消息为："+ intent.getStringExtra("msg"),
10	Toast.LENGTH_SHORT).show();
11	}
12	}

在界面中添加两个按钮，分别用于发送一个普通广播和发送一个有序广播，XML 文件对应的代码如表 6-9 所示。

表 6-9　XML 文件对应的代码

序号	代　　码
01	<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	xmlns:tools="http://schemas.android.com/tools"
03	……
04	android:orientation="horizontal">
05	<Button
06	android:id="@+id/btnBasicSendNormal"
07	android:layout_width="wrap_content"
08	android:layout_height="wrap_content"
09	android:onClick="start"
10	android:text="发送一个普通广播"　/>
11	<Button
12	android:id="@+id/btnBasicSendOrdered"
13	android:layout_width="wrap_content"
14	android:layout_height="wrap_content"
15	android:onClick="stop"
16	android:text="发送一个有序广播"　/>
17	</LinearLayout>

接着声明一个 Activity 子类 BasicActivity，用于发送 Broadcast，其代码如表 6-10 所示。

表 6-10　类 BasicActivity 对应的代码

序号	代　　码
01	package com.example.exampleapp0603;

续表

序号	代　码
02	import android.app.Activity;
03	import android.os.Bundle;
04	import android.view.Menu;
05	import android.view.MenuItem;
06	import android.content.Intent;
07	import android.view.View;
08	import android.widget.Button;
09	public class MainActivity extends Activity {
10	Button btnNormal, btnOrdered;
11	@Override
12	protected void onCreate(Bundle savedInstanceState) {
13	super.onCreate(savedInstanceState);
14	setContentView(R.layout.activity_main);
15	btnNormal = (Button) findViewById(R.id.btnBasicSendNormal);
16	btnOrdered = (Button) findViewById(R.id.btnBasicSendOrdered);
17	btnNormal.setOnClickListener(new View.OnClickListener() {
18	@Override
19	public void onClick(View v) {
20	Intent broadcast=new Intent();
21	broadcast.setAction("com.example.exampleapp0603.broadcast");
22	broadcast.putExtra("msg", "这是一个普通广播");
23	sendBroadcast(broadcast);
24	}
25	});
26	btnOrdered.setOnClickListener(new View.OnClickListener() {
27	@Override
28	public void onClick(View v) {
29	Intent broadcast=new Intent();
30	broadcast.setAction("com.example.exampleapp0603.broadcast");
31	broadcast.putExtra("msg", "这是一个有序广播");
32	sendOrderedBroadcast(broadcast, null);
33	}
34	});
35	}

由表 6-10 所示的代码可知，发送 Broadcast 时，首先在需要发送信息的地方，把要发送的信息装入一个 Intent 对象，然后通过调用 sendBroadcast()或 sendOrderBroadcast()方法，把 Intent 对象以广播方式发送出去。

当 Intent 发送以后，所有已经注册的 BroadcastReceiver 会检查注册时的 IntentFilter 是否与发送的 Intent 相匹配，若匹配则就会调用 BroadcastReceiver 的 onReceive()方法。所以当我们定义一个 BroadcastReceiver 时，都需要实现 onReceive()方法。

在创建完我们的 BroadcastReceiver 之后，还不能够使它进入工作状态，我们需要为它注册一个指定的广播地址。没有注册广播地址的 BroadcastReceiver 就像一个缺少选台按钮的收音机，虽然功能具备，但也无法收到电台的信号。在清单文件 AndroidManifest.xml 中注册 Intent 对象，其代码如下所示。

```
<receiver android:name="com.example.exampleapp0603.BasicBroadcast">
    <intent-filter>
        <action android:name="com.example.exampleapp0603.broadcast"/>
    </intent-filter>
</receiver>
```

运行 BroadcastReceiver 使用实例程序，显示如图 6-5 所示的界面。



图 6-5　BroadcastReceiver 使用实例程序的运行效果

在运行界面中单击【发送一个普通广播】按钮，显示如图 6-6 所示的提示信息，单击【发送一个有序广播】显示如图 6-7 所示的提示信息。



图 6-6　单击【发送一个普通广播】按钮时显示的提示信息



图 6-7　单击【发送一个有序广播】时显示的提示信息



【任务 6-1】获取系统的唤醒服务

【任务描述】

我们使用手机浏览网页时，如果在一段时间内没有触屏，屏幕就会黑屏并锁屏了，编写程序实现程序开始时得到系统的唤醒服务，程序退出时关闭程序的唤醒服务。

通常情况下，在一个应用 Activity 的 onresume 方法中获得屏幕锁，在 onstop 方法中释放屏幕锁，本任务为简化程序，方便演示系统的唤醒服务和电源服务，要求使用按钮来进行屏幕锁的获得和释放。

【知识索引】

（1）WakeLock 类及 acquire()、isHeld()、release()等方法。

（2）PowerManager 类、Context 类，getSystemService()方法、getCanonicalName()方法、POWER_SERVICE 常量、FULL_WAKE_LOCK 常量。

（3）Toast 类及 makeText()、show()方法。

【实施过程】

1．创建 Android 项目 App0601

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0601。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 2 个 Button 控件，将 Button 控件的 id 属性分别设置为 btnGet 和 btnRelease、text 属性分别设置为“获取唤醒锁”和“释放唤醒锁”。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示。

```
private WakeLock wakeLock;          //得到唤醒锁对象
private Button getLock;             //声明获取唤醒锁对象的按钮
private Button releaseLock;         //声明释放唤醒锁对象的按钮
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取 Button 对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 6-11 所示。

表 6-11　onCreate()方法的代码

序号	代　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	getLock = (Button)findViewById(R.id.btnGet);
06	releaseLock = (Button)findViewById(R.id.btnRelease);
07	//设置对象的监听器
08	setListener();
09	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()的现代码如下所示。

```
getLock.setOnClickListener(mylistener);
releaseLock.setOnClickListener(mylistener)
```

自定义 OnClickListener 类对象的代码如表 6-12 所示，当用户单击获得唤醒按钮时调用 acquireWakeLock()方法，当用户单击释放唤醒锁按钮时调用 releaseWakeLock()方法。

表 6-12　自定义 OnClickListener 类对象的代码

序号	代　码
01	OnClickListener mylistener = new OnClickListener() {
02	@Override
03	public void onClick(View v) {
04	// TODO Auto-generated method stub
05	switch (v.getId()) {
06	case R.id.btnGet:

续表

序号	代　　码
07	acquireWakeLock();　　// 开始获得唤醒锁
08	break;
09	case R.id.btnRelease:
10	releaseWakeLock();　　// 释放锁
11	break;
12	default:
13	break;
14	}
15	}
16	};

自定义方法 acquireWakeLock()的代码如表 6-13 所示，第 06 行代码通过 getSystemService 对象得到系统的 PowerService。

表 6-13　自定义方法 acquireWakeLock()的代码

序号	代　　码
01	// 开始获得唤醒锁
02	private void acquireWakeLock() {
03	if (wakeLock == null) {
04	Toast.makeText(this, "得到唤醒锁", Toast.LENGTH_SHORT).show();
05	// 得到电源管理服务
06	PowerManager pm = (PowerManager)getSystemService(Context.POWER_SERVICE);
07	// 加锁控制电源的状态
08	wakeLock = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK,
09	this.getClass().getCanonicalName());
10	// 获取相应的锁，屏幕将停留在设定的状态，一般为亮、暗状态
11	wakeLock.acquire();
12	}
13	}

自定义方法 releaseWakeLock()的代码如表 6-14 所示，第 04 行调用唤醒锁对象 wakeLock 的 release()方法释放屏幕锁的状态。

表 6-14　自定义方法 releaseWakeLock()的代码

序号	代　　码
01	private void releaseWakeLock() {
02	if (wakeLock != null && wakeLock.isHeld()) {
03	// 释放掉正在运行的 CPU 或关闭屏幕
04	wakeLock.release();
05	wakeLock = null;
06	}
07	Toast.makeText(MainActivity.this, "释放唤醒锁",Toast.LENGTH_SHORT).show();
08	}

4．在 AndroidManifest.xml 文件中增加获得系统唤醒锁的权限

由于本任务得到了系统的电源服务，所以需要在 AndroidManifest.xml 文件输入如下所示的代码，增加获得系统唤醒锁的权限，以保证程序可以获得唤醒锁权限。

```
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

5．程序运行与功能测试

Android 项目 App0601 的初始运行状态如图 6-8 所示。

单击【获得唤醒锁】按钮，得到系统的唤醒锁，显示如图 6-9 所示的“得到唤醒锁”提示信息，此时系统不会进入休眠和锁屏。

单击【释放唤醒锁】按钮，释放系统的唤醒锁，显示如图 6-10 所示的“释放唤醒锁”提示信息，此时系统经过默认的时间屏幕就会进入黑屏锁屏状态。



图 6-8　Android 项目 App0601 的初始运行状态



图 6-9　“得到唤醒锁”的提示信息

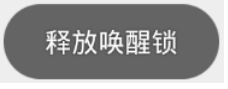


图 6-10　“释放唤醒锁”的提示信息

【任务 6-2】获取系统的屏蔽状态

【任务描述】

系统的屏幕状态一般分为正常、锁屏和休眠，程序锁屏的时候，会暂时关闭某些服务，但有些应用（例如玩游戏时）不能让系统进入休眠状态，所以当用户进行某项应用操作时，需要得到和判断屏幕状态。编写程序通过系统服务得到系统的屏幕状态，通过单击按钮显示屏幕状态。

【知识索引】

（1）KeyguardManager 类、getSystemService()方法、isKeyguardLocked()方法。

（2）Toast 类、makeText()和 show()方法、LENGTH_SHORT 和 LENGTH_SHORT 常量。

【实施过程】

1．创建 Android 项目 App0602

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0602。

2．完善布局文件 activity_main.xml 与界面设计

界面的布局方式为将默认的相对布局方式（RelativeLayout），将默认添加的 TextView 控件删除，然后添加 1 个 Button 控件，将 Button 控件的 id 属性设置为 btnGetStatus、text 属性设置为“@string/getStatus”。

变量 getStatus 在 string.xml 文件中的声明代码如下所示。

```
<string name="getStatus">查看屏幕的状态</string>
```

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private Button btnKeyguard;          //声明锁屏按钮对象
private KeyguardManager km;          //声明 KeyguardManager 对象（键盘管理对象）
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获得 KeyguardManager 服务和获取布局中的 Button 对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 6-15 所示。

表 6-15　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 获得 KeyguardManager 服务
06	km=(KeyguardManager)getSystemService(Context.KEYGUARD_SERVICE);
07	// 得到布局中的 Button 对象
08	btnKeyguard = (Button) findViewById(R.id.btnGetStatus);
09	//设置对象的监听器
10	setListener();
11	}

表 6-15 中第 06 行通过 getSystemService 对象得到系统的锁屏键的服务。

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()实现代码如表 6-16 所示。

表 6-16　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	// 设置对象的监听器
03	btnKeyguard.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View arg0) {
06	//判断当前屏幕的状态
07	if(km.isKeyguardLocked())
08	{
09	Toast.makeText(MainActivity.this, "锁屏", Toast.LENGTH_SHORT).show();
10	}
11	else
12	{

续表

序号	代　　码
13	Toast.makeText(MainActivity.this, "没有锁屏", Toast.LENGTH_SHORT).show();
14	}
15	}
16	});
17	}

表 6-16 中第 07 行通过 KeyguardManager 对象 km 的 isKeyguardLocked()方法得到屏幕是否为锁屏状态，然后显示对应的提示信息。

4．在 AndroidManifest.xml 文件中增加获得系统的锁屏权限

由于本任务通过得到系统的锁屏服务进行屏幕状态的查看，需要在 AndroidManifest.xml 文件输入如下所示的代码，增加获得系统的锁屏权限，以保证程序可以获得系统的锁屏权限。

```
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
```

5．程序运行与功能测试



图 6-11　Android 项目 App0602 的初始运行状态

Android 项目 App0602 的初始运行状态如图 6-11 所示。

单击【查看屏幕的状态】按钮，显示“没有锁屏”的提示信息，因为此时没有进行锁屏操作。

【任务 6-3】 获取当前网络状态

【任务描述】

编写程序获取并查看当前网络状态，以便根据网络状态实现相应的操作。

【知识索引】

（1）ConnectivityManager 类、getSystemService()方法、getActiveNetworkInfo()方法、getNetworkInfo()方法、getState()方法、getNetworkInfo()方法、TYPE_MOBILE 常量、TYPE_WIFI 常量。

（2）NetworkInfo 类、isAvailable()方法。

（3）State 类、CONNECTED 常量。

【实施过程】

1．创建 Android 项目 App0603

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0603。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），默认状态已添加 1 个 TextView 控件，再添加 1 个 Button 控件，将 TextView 控件的 id 属性设置为 btnNetwork、text 属性设置为“显示当前网络状态”，将 Button 控件的 id 属性设置为 tvNetworkshow。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private Button btn;                          // 声明显示当前网络状态的按钮
private TextView tv;                         //定义显示当前网络连接状态的文本框
ConnectivityManager connManager ;            //声明网络连接管理器
private NetworkInfo networkInfo;             //声明代表联网状态的 NetworkInfo 对象
private String strNetWork;                   //定义当前网络连接的 String
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取 ConnectivityManager 对象和所有控件对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 6-17 所示。

表 6-17　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 获得网络连接服务管理器对象
06	connManager=(ConnectivityManager)getSystemService(Context.CONNECTIVITY_SERVICE);
07	// 获取代表联网状态的 NetWorkInfo 对象（当前的网络连接信息）
08	networkInfo = connManager.getActiveNetworkInfo();
09	// 得到布局中的所有对象
10	btn = (Button) findViewById(R.id.btnNetwork);
11	tv=(TextView)findViewById(R.id.tvNetworkshow);
12	//设置对象的监听器
13	setListener();
14	}

（4）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()实现代码如表 6-18 所示。

表 6-18　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	// 设置对象的监听器
03	btn.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View arg0) {
06	//当前网络是否可用
07	if(networkInfo.isAvailable())
08	{

续表

序号	代　　码
09	strNetWork="当前网络可用\n";
10	}
11	else{
12	strNetWork="当前网络不可用\n";
13	}
14	//获取 GPRS 网络模式连接的描述
15	State state = connManager.getNetworkInfo(ConnectivityManager.TYPE_MOBILE).getState();
16	//State.CONNECTED 表示当前 GPRS 已连接
17	if(state==State.CONNECTED)
18	{
19	strNetWork+="GPRS 网络已连接\n";
20	}
21	//获取 WIFI 网络模式连接的描述
22	state = connManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI).getState();
23	//State.CONNECTED 表示当前 WIFI 已连接
24	if(state==State.CONNECTED)
25	{
26	strNetWork+="WIFI 网络已连接\n";
27	}
28	//设置 textview 的 text 属性
29	tv.setText(strNetWork);
30	}
31	});
32	}

表 6-18 中第 07 行代码通过 isAvailable()方法得到 NetworkInfo 对象是否可用，然后在第 15 行得到当前已连接的网络类型，在第 17 行和第 24 行分别判断当前网络的连接类型，并且构造提示信息字符串在 TextView 控件中显示相应的信息。

4．在 AndroidManifest.xml 文件中增加获得系统的网络状态权限

在 AndroidManifest.xml 文件中输入如下所示的代码，增加获得系统网络状态权限的代码，以保证程序可以获得系统的网络状态权限。

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

5．程序运行与功能测试

Android 项目 App0603 的初始运行状态如图 6-12 所示。

单击【显示当前网络状态】按钮，显示如图 6-13 所示的当前网络状态提示信息。



图 6-12　Android 项目 App0603 的初始运行状态

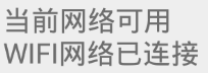


图 6-13　当前网络状态的提示信息

【任务 6-4】 实现音量控制

【任务描述】

编写程序获取系统的音量管理器服务，然后得到相应的音量参数，调节系统音量，并在屏幕显示当前的音量大小。

【知识索引】

（1）AudioManager 类、getSystemService()方法、getStreamVolume()方法，STREAM_MUSIC、STREAM_VOICE_CALL 、 STREAM_SYSTEM 、 STREAM_ALARM 、 STREAM_RING 、ADJUST_LOWER、ADJUST_RAISE、ADJUST_SAME、FX_FOCUS_NAVIGATION_UP 等常量。

（2）Context 类、AUDIO_SERVICE 常量。

【实施过程】

1．创建 Android 项目 App0604

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0604。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 3 个 Button 控件和 1 个 TextView 控件，将 Button 控件的 id 属性分别设置为 btn_currentvolume、btn_increaselvolume、btn_declinevolume，将 Button 控件的 text 属性分别设置为“显示当前音量”、“增大系统音量”、“减小系统音量”，将 TextView 控件的 id 属性设置为 tv_volumn。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private Button btnCurrentVolume;      // 定义当前音量的按钮
private TextView tvVolume;            // 定义显示当前音量的 TextView 控件
private Button btnIncreaseVolume;     // 定义增加音量的按钮
private Button btnDeclineVolume;      // 定义减小音量的按钮
private AudioManager audioManager ;   //定义 AudioManager 对象（音频管理器对象）
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取 AudioManager 对象和所有控件对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 6-19 所示。

表 6-19　onCreate()方法的代码

序号	代　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	//取得 AudioManager 对象
06	audioManager = (AudioManager)getSystemService(Context.AUDIO_SERVICE);

续表

序号	代　　码
07	//得到布局中的所有控件对象
08	btnCurrentVolume = (Button) findViewById(R.id.btn_currentvolume);
09	btnDeclineVolume=(Button)findViewById(R.id.btn_declinevolume);
10	btnIncreaseVolume=(Button)findViewById(R.id.btn_increaselvolume);
11	tvVolume=(TextView)findViewById(R.id.tv_volumn);
12	//设置对象的监听器
13	setListener();
14	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()的实现代码如表 6-20 所示。

表 6-20　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	//设置对象的监听器
02	private void setListener() {
03	btnCurrentVolume.setOnClickListener(listener);
04	btnDeclineVolume.setOnClickListener(listener);
05	btnIncreaseVolume.setOnClickListener(listener);
06	}

自定义 OnClickListener 类对象的代码如表 6-21 所示，当单击某个按钮时执行相应的代码。

表 6-21　自定义 OnClickListener 类对象的代码

序号	代　　码
01	OnClickListener listener=new OnClickListener() {
02	@Override
03	public void onClick(View v) {
04	switch (v.getId()) {
05	//当前的音量
06	case R.id.btn_currentvolume:
07	//得到当前的通话音量
08	int currentCall=audioManager.getStreamVolume(AudioManager.STREAM_VOICE_CALL);
09	//得到当前的系统音量
10	int currentSystem = audioManager.getStreamVolume(AudioManager.STREAM_SYSTEM);
11	//得到当前的音乐音量
12	int currentMusic = audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
13	//得到当前的提示声音音量
14	int currentTip = audioManager.getStreamVolume(AudioManager.STREAM_ALARM);
15	tvVolume.setText("当前的通话音量："+currentCall+"\n 当前的系统音量: "
16	+currentSystem+"\n 当前的音乐音量: "+currentMusic
17	+"\n 当前的提示声音音量: "+currentTip);

续表

序号	代　　码
18	break;
19	//增加系统音量
20	case R.id.btn_increaselvolume:
21	audioManager.adjustStreamVolume(AudioManager.STREAM_SYSTEM,
22	AudioManager.ADJUST_RAISE, AudioManager.FX_FOCUS_NAVIGATION_UP);
23	//参数 1：声音类型，可取为 STREAM_VOICE_CALL（通话）、
24	//STREAM_SYSTEM（系统声音）、STREAM_RING（铃声）、
25	//STREAM_MUSIC（音乐）、STREAM_ALARM（闹铃声）
26	//参数 2：调整音量的方向，可取 ADJUST_LOWER（降低）、
27	//ADJUST_RAISE（升高）、ADJUST_SAME
28	//参数 3：可选的标志位
29	break;
30	//减小系统音量
31	case R.id.btn_declinevolume:
32	audioManager.adjustStreamVolume(AudioManager.STREAM_SYSTEM,
33	AudioManager.ADJUST_LOWER,AudioManager.FX_FOCUS_NAVIGATION_UP);
34	break;
35	}
36	}
37	};

当单击【显示当前音量】按钮时，调用表 6-21 中第 06～18 行代码，得到 AudioManager 对象的各种声音参数；当单击【增大系统音量】按钮时，调用表 6-21 中第 20～29 行代码，通过 AudioManager 对象增加系统音量；当单击【减小系统音量】按钮时，调用表 6-21 中第 31～34 行代码，通过 AudioManager 对象降低系统音量。

4．程序运行与功能测试

Android 项目 App0604 的初始运行状态如图 6-14 所示。



图 6-14　Android 项目 App0604 的初始运行状态

单击【显示当前音量】按钮，显示与音量相关的信息如图 6-15 所示。

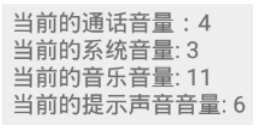


图 6-15　与音量相关的信息显示

单击【增大系统音量】或【减小系统音量】，然后再次单击【显示当前音量】按钮，在显示的音量信息中，当前的系统音量大小会发生变化。

【任务 6-5】实现程序开机自动启动

【任务描述】

对于某些手机软件可以设置为开机时自动启动，编写程序启动后台服务或广播监听器，监控手机启动。

【知识索引】

（1）BroadcastReceiver、OnClickListener、IntentFilter、BootCompletedReceiver 类。

（2）Intent 类、addFlag()方法、ACTION_BOOT_COMPLETED 和 FLAG_ACTIVITY_NEW_TASK 常量

（3）Context 类、startActivity()方法、registerReceiver()方法、unregisterReceiver()方法。

【实施过程】

1．创建 Android 项目 App0605

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0605。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 2 个 Button 控件，将 Button 控件的 id 属性分别设置为 btn_startafterboot、btn_notstartafterboot，text 属性分别设置为“开机启动”、“开机不启动”。

3．自定义广播接收器类 BootCompletedReceiver 与实现程序功能

本任务需要自定义一个广播接收器类，用来接收系统重启的广播，自定义 BootCompletedReceiver 类的代码如表 6-22 所示。

表 6-22　自定义 BootCompletedReceiver 类的代码

序号	代　　码
01	import android.content.BroadcastReceiver;
02	import android.content.Context;
03	import android.content.Intent;
04	// 自定义广播接收器类
05	public class BootCompletedReceiver extends BroadcastReceiver {
06	@Override
07	public void onReceive(Context arg0, Intent arg1) {
08	// 检测手机是否是重启广播
09	if (arg1.getAction().equals(Intent.ACTION_BOOT_COMPLETED)) {
10	// 检测到手机启动后，启动 MainActivity

续表

序号	代　　码
11	Intent newIntent = new Intent(arg0, MainActivity.class);
12	// 让 activity 启动一个新任务， 注意，必须添加这个标记，否则启动会失败
13	newIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
14	// 启动 MainActivity
15	arg0.startActivity(newIntent);
16	}
17	}
18	}

4．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private Button btnStartAfterBoot;          // 声明开机启动的按钮
private Button btnNotStartAfterBoot;       // 声明开机不启动按钮
//声明自定义广播接收器类 BootCompletedReceiver 的对象
private BootCompletedReceiver receiver;
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取 Button 对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 6-23 所示。

表 6-23　onCreate()方法的代码

序号	代　　码
01	@Override
02	public void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	//给 BootCompletedReceiver 对象赋值
06	receiver=new BootCompletedReceiver();
07	//得到布局中的所有对象
08	btnNotStartAfterBoot = (Button) findViewById(R.id.btn_notstartafterboot);
09	btnStartAfterBoot=(Button)findViewById(R.id.btn_startafterboot);
10	//设置对象的监听器
11	setListener();
12	}

（3）编写代码为按钮自定义单击监听器

编写代码自定义单击监听器 OnClickListener，方法 setListener()的实现代码如表 6-24 所示。

表 6-24　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	//设置对象的监听器

续表

序号	代　　码
02	private void setListener() {
03	btnStartAfterBoot.setOnClickListener(listener);
04	btnNotStartAfterBoot.setOnClickListener(listener);
05	}

自定义 OnClickListener 类的代码如表 6-25 所示。

表 6-25　自定义 OnClickListener 类的代码

序号	代　　码
01	OnClickListener listener=new OnClickListener() {
02	@Override
03	public void onClick(View v) {
04	switch (v.getId()) {
05	//注册广播
06	case R.id.btn_startafterboot:
07	//注册监听者，第一个参数是需要绑定的监听器，第二个是需要监听的广播
08	Context.registerReceiver(receiver,
09	new IntentFilter("android.intent.action.BOOT_COMPLETED"));
10	Toast.makeText(MainActivity.this, "开机自启动已完成",
11	Toast.LENGTH_SHORT).show();
12	break;
13	case R.id.btn_notstartafterboot:
14	if(receiver!=null)
15	{
16	//取消监听者
17	Context.unregisterReceiver(receiver);
18	Toast.makeText(MainActivity.this, "取消开机自启动已完成",
19	Toast.LENGTH_SHORT).show();
20	}
21	break;
22	}
23	}
24	};

单击【开机启动】按钮时，调用表 6-25 中第 08～11 行代码，第 08 行注册广播接收器，单击【开机不启动】按钮时，调用表 6-25 中第 17～19 行代码，第 17 行取消此广播接收器。

5．在 AndroidManifest.xml 文件中增加获得系统重启广播权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加获得系统重启广播的权限。

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
```

6．程序运行与功能测试

Android 项目 App0605 的初始运行状态如图 6-16 所示。



图 6-16　Android 项目 App0605 的初始运行状态

单击【开机启动】按钮，显示如图 6-17 所示的“开机自启动已完成”的提示信息，单击【开机不启动】按钮，显示如图 6-18 所示的“取消开机自启动已完成”的提示信息。



图 6-17　“开机自启动已完成”的提示信息



图 6-18　“取消开机自启动已完成”的提示信息

【任务 6-6】监控手机电池电量

【任务描述】

编写程序监控手机电池的电量，通过广播接收器接收电池电量低的广播，当电池电量偏低时自动提醒。

【知识索引】

（1）BroadcastReceiver 类、PowerLowReceiver 类、IntentFilter 类。

（2）Intent 类、ACTION_BATTERY_LOW 常量。

（3）Context 类、registerReceiver()方法、unregisterReceiver()方法。

【实施过程】

1．创建 Android 项目 App0606

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0606。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 2 个 Button 控件，将 Button 控件的 id 属性分别设置为 btn_startpowerlow、btn_cancelpowerlow，text 属性分别设置为“启动电池电量低的广播”、“取消电池电量低的广播”。

3．自定义广播接收器类 BootCompletedReceiver 与实现程序功能

本任务需要自定义一个广播接收器类，用来接收电池电量低的广播，自定义 BootCompletedReceiver 类的代码如表 6-26 所示。

表 6-26　自定义 BootCompletedReceiver 类的代码

序号	代　码
01	import android.content.BroadcastReceiver;
02	import android.content.Context;

续表

序号	代　　码
03	import android.content.Intent;
04	import android.widget.Toast;
05	//定义接收系统电量低的广播接收器类
06	public class PowerLowReceiver extends BroadcastReceiver {
07	@Override
08	public void onReceive(Context arg0, Intent arg1) {
09	// Intent.ACTION_BATTERY_LOW;
10	// 表示电池电量低
11	if (arg1.getAction().equals(Intent.ACTION_BATTERY_LOW)) {
12	Toast.makeText(arg0, "检测到电池电量低的广播", Toast.LENGTH_SHORT).show();
13	}
14	}
15	}

4．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private Button btnPowerLow;                    // 声明电池电量低广播的按钮
private Button btnCancelPowerLow;              // 声明取消电池电量低广播的按钮
private PowerLowReceiver powerLowReceiver;     // 声明自定义广播接收器类对象
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取 Button 对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 6-27 所示。

表 6-27　onCreate()方法的代码

序号	代　　码
01	@Override
02	public void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 给自定义广播接收器类 PowerLowReceiver 对象赋值
06	powerLowReceiver = new PowerLowReceiver();
07	// 得到布局中的所有对象
08	btnPowerLow = (Button) findViewById(R.id.btn_startpowerlow);
09	btnCancelPowerLow = (Button) findViewById(R.id.btn_cancelpowerlow);
10	// 设置对象的监听器
11	setListener();
12	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()实现代码如表 6-28 所示。

表 6-28　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	// 设置对象的监听器
02	private void setListener() {
03	btnPowerLow.setOnClickListener(listener);
04	btnCancelPowerLow.setOnClickListener(listener);
05	}

自定义 OnClickListener 类的代码如表 6-29 所示。

表 6-29　自定义 OnClickListener 类的代码

序号	代　　码
01	OnClickListener listener = new OnClickListener() {
02	@Override
03	public void onClick(View v) {
04	switch (v.getId()) {
05	// 开启电量低的接收广播
06	case R.id.btn_startpowerlow:
07	// 注册监听者，第一个参数是需要绑定的监听器，第二个是需要监听的广播
08	Context.registerReceiver(powerLowReceiver, new IntentFilter(
09	Intent.ACTION_BATTERY_LOW));
10	Toast.makeText(MainActivity.this, "设置电池电量低广播启动已完成",
11	Toast.LENGTH_SHORT).show();
12	break;
13	// 取消监听者
14	case R.id.btn_cancelpowerlow:
15	if (powerLowReceiver != null) {
16	// 取消监听者
17	Context.unregisterReceiver(powerLowReceiver);
18	Toast.makeText(MainActivity.this, "取消电池电量低广播已完成",
19	Toast.LENGTH_SHORT).show();
20	}
21	break;
22	}
23	}
24	};

单击【启动电池电量低的广播】按钮时，调用表 6-29 中第 08～12 行代码，第 08～09 行注册广播接收器，单击【取消电池电量低的广播】按钮时，调用表 6-29 中第 17～19 行代码，第 17 行取消此广播接收器。

5．程序运行与功能测试

Android 项目 App0606 的初始运行状态如图 6-19 所示。

单击【启动电池电量低的广播】按钮，开启广播监听，显示如图 6-20 所示的“设置电池

电量低广播启动已完成”的提示信息，单击【取消电池电量低的广播】按钮，取消对应的广播接收器，显示如图 6-21 所示的“取消电池电量低广播已完成”的提示信息。



图 6-19　Android 项目 App0606 的初始运行状态



图 6-20　“设置电池电量低广播启动已完成”的提示信息



图 6-21　“取消电池电量低广播已完成”的提示信息

单元小结

Service（服务）和 BroadcastReceiver（广播接收者）都是 Android 系统中的重要组件。Service（服务）是一个没有用户界面在后台运行的应用组件，并且可以和其他组件进行交互。BroadcastReceiver 是用来接收来自系统和应用中的广播，Broadcast 是 Android 中一种广泛应用在应用程序之间传输信息的机制，而 BroadcastReceiver 是对发送出来的 Broadcast 进行过滤接受并响应的一类组件。BroadcastReceiver 是一个系统全局的监听器，用于监听系统全局的 Broadcast 消息，可以很方便地实现系统组件之间的通信。本单元主要阐述了 Service 的基本概念、创建与注册、启动流程、生命周期、类型等内容，BroadcastReceiver 的基本概念、生命周期、注册方式、类型等内容。通过多个实例学会了 Service 和 BroadcastReceiver 的典型应用。

单元习题

一、填空题

1．在创建服务时，必须要继承（　　）类，绑定服务时，必须要实现服务的（　　）方法。

2．服务的开启方式有两种，分别是（　　）和（　　）。

3．在清单文件中，注册服务时应该使用的结点为（　　）。

4．广播接收者有清单文件，使用（　　）注册。终止广播需要使用（　　）方法。

5．广播的发送有两种形式，分别为（　　）和（　　）。

6．代码注册广播需要使用（　　）方法，解除广播需要使用（　　）方法。

二、选择题

1．每一次启动服务都会调用（　　）方法。

A．onCreate()　　B．onStart()　　C．onResume()　　D．onStartCommand()

2．下列方法中，不属于 Service 生命周期是（　　）。

A．onResume()　　B．onStart()　　C．onStop()　　D．onDestory()

3．继承 BroadcastReceiver 会重写（　　）方法。

A．onReceiver()　　B．onUpdate()　　C．onCreate()　　D．onStart()

4．关于广播的作用，说法正确的是（　　）。

A．它主要用来接收系统发布的一些消息

B．它可以进行耗时的操作

C．它可以启动一个 Activity

D．它可帮助 Activity 修改用户界面

5．下列方法中，用于发送一条有序广播的方法是（　　）。

A．startBroadcastRceiver()　　B．sendOrderedBroadcast()

C．sendBroadcast()　　D．sendReceiver()

6．在清单文件中，注册广播时使用的结点是（　　）。

A．<activity>　　B．<broadcast>　　C．<receiver>　　D．<broadcastreceiver>

7．BroadcastReceiver 说明错误的是（　　）。

A．是用来接收广播 Intent 的

B．一个广播 Intent 只能被一个订阅了此广播的 BroadcastReceiver 所接收

C．对有序广播，系统会根据接收者声明的优先级别按顺序逐个执行接收者

D．接收者声明的优先级别在 android:priority 属性中声明，数值越大优先级别越高

8．下列选项中，属于绑定服务特点的是（　　）。（多选题）

A．以 bindService()方法开启　　B．调用者关闭后服务关闭

C．必须实现 ServiceConnection()　　D．使用 stopService()方法关闭服务

9．Service 与 Activity 的共同点是（　　）。（多选题）

A．都是 4 大组件之一　　B．都有 onResume()方法

C．都可以被远程调用　　D．都可以自定义美观界面

10．关于 Service 生命周期的 onCreate()和 onStart()方法，正确的是（　　）。（多选题）

A．如果 Service 已经启动，将先后调用 onCreate()和 onStart()方法

B．当第一次启动的时候先后调用 onCreate()和 onStart()方法

C．当第一次启动的时候只会调用 onCreate()方法

D．如果 Service 已经启动，只会执行 onStart()方法，不再执行 onCreate()方法

三、简答题

1．简要说明 Service 的几种启动方式及其特点。

2．简要说明 Service 常用的生命周期回调方法。

3．简要说明注册广播有几种方式，这些方式各有何优缺点。

4．简要说明接收系统广播时哪些功能需要使用权限。

单元 7

Android 的网络与通信应用程序设计

Android 提供了多种方式实现网络通信，包括 Socket 通信、HTTP 通信、URL 通信和 WebView，其中最常用的是 HTTP 通信。Android 客户端访问网络发送 HTTP 请求的方式一般有两种：HttpURLConnection 和 HttpClicent。HttpURLConnection 是 Java 的标准类，继承自 URLConnection，可用于向指定网站发送 GET 请求或 POST 请求。HttpClient 是一个开源项目，是 Apache Jakarta Common 下的一个子项目，可以用来提供高效的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。



教学导航

教学目标	（1）理解 HTTP 协议、URL 请示的类别 （2）理解并掌握 Android 的线程与 Hander 消息机制 （3）学会使用 HttpURLConnection 访问网络 （4）学会使用 HttpClient 访问网络
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	6 课时



知识导读

7.1 HTTP 协议简介

我们在 Internet 网搜索资源时，通常先打开百度主页，然后输入关键词查找所需资源。在浏览器的地址栏中输入百度的网址（www.baidu.com）单击【搜索】按钮，就会进入百度主页。这个访问百度的过程就是通过 HTTP 协议完成的，HTTP（Hyper Text Transfer Protocol）即超文本传输协议，它规定了浏览器和万维网服务器之间相互通信的规则。

当客户端在与服务器建立连接后，向服务器发送的请求，被称作 HTTP 请求。服务器端接收到请求后会做出响应，称为 HTTP 响应。手机终端访问服务器的示意图如图 7-1 所示，该图展示了 HTTP 协议的通信过程。

从图 7-1 可以看出，使用手机客户端访问百度网站时，会发送一个 HTTP 请求。当服务器端接收到这个请求后，会做出响应并将百度首页返回给客户端浏览器。这个请求和响应的过程实际上就是 HTTP 通信的过程。



图 7-1　手机终端访问服务器的示意图

7.2 URL 请求的类别

URL 请求的类别分为两类：GET 请求与 POST 请求。二者的区别在于：

① GET 请求可以获取静态页面，也可以把参数放在 URL 字串后面进行传递。使用 GET 方式访问网络时 URL 的长度是有限制的。

② POST 方式用来向目的服务器发出请求，其参数不是放在 URL 字串里面，而是放在 HTTP 请求的正文内，并且 POST 方式对 URL 的长度没有限制。

7.3 Android 的线程与 Hander 消息机制

出于性能优化的考虑，Android UI（User Interface，用户界面）操作并不是线程安全，如果有多个线程并发操作 UI 组件，可能导致线程安全问题。可以设想下，如果在一个 Activity 中有多个线程去更新 UI，并且都没有加锁机制，可能会导致什么问题？产生界面混乱，如果加锁的话可以避免该问题但又会导致性能下降。因此 Android 规定只允许 UI 线程修改 Activity 的 UI 组件。当程序第一次启动时，Android 会同时启动一条主线程(Main Thread)，主线程主要负责处理与 UI 相关的事件，例如用户按钮事件，并把相关的事件分发到对应的组件进行处理，因此主线程又称为 UI 线程。那么怎么在新启动的线程中更新 UI 组件呢，这就需要借助 handler 的消息传递机制来实现了。

刚刚开始接触 Android 线程编程时，习惯像 Java 一样，试图用下面的代码解决问题：

```
new Thread( new Runnable() {
   public void run() {
       myView.invalidate();
    }
}).start();
```

这些代码虽然也可以刷新 UI 界面，但是它违背了单线程模型，Android 中不推荐使用。Android UI 操作并不是线程安全的，并且这些操作必须在 UI 线程中执行。

在 Android 平台中需要反复按周期执行方法，可以使用 Java 上自带的 TimerTask 类，TimerTask 类相对于 Thread 来说消耗资源更低，除了使用 Android 自带的 AlarmManager，使

用 Timer 定时器也是一种更好的解决方法，示例代码如下。

```
public class JavaTimer extends Activity {
    Timer timer = new Timer();
    TimerTask task = new TimerTask(){
        public void run() {
            setTitle("hear me?");
        }
    };
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
         timer.schedule(task, 10000);
    }
}
```

实际上这样做也是不行的，这跟 Android 的线程安全有关。

正确的实现线程方法是使用 Thread+Handler 机制，即利用 Handler 来实现 UI 线程的更新。Thread 线程发出 Handler 消息，通知更新 UI。Handler 根据接收的消息，处理 UI 更新。实现代码如下。

```
Handler myHandler = new Handler() {
     public void handleMessage(Message msg) {
         switch (msg.what) {
             case TestHandler.GUIUPDATEIDENTIFIER:
                 myBounceView.invalidate();
                 break;
         }
         super.handleMessage(msg);
     }
 };
class MyThread implements Runnable {
     public void run() {
         while (!Thread.currentThread().isInterrupted()) {
             Message message = new Message();
             message.what = TestHandler.GUIUPDATEIDENTIFIER;
             TestHandler.this.myHandler.sendMessage(message);
             try {
                 Thread.sleep(100);
             } catch (InterruptedException e) {
                 Thread.currentThread().interrupt();
             }
         }
     }
 }
```

我们创建的 Service、Activity 以及 Broadcast 均是一个主线程处理，这里我们可以理解为 UI 线程。但是在操作一些耗时操作时，例如 I/O 读写大文件、数据库操作以及网络下载等操作都需要很长时间，为了不阻塞用户界面，此时我们可以考虑使用 Thread 线程来解决。

对于从事过 J2ME 开发的程序员来说 Thread 比较简单，直接匿名创建重写 run 方法，调用 start 方法执行即可，或者从 Runnable 接口继承。但对于 Android 平台来说 UI 控件都没有设计成为线程安全类型，所以需要引入一些同步的机制来使其刷新。

对于线程中刷新一个 View 为基类的界面，可以使用 postInvalidate()方法在线程中来处理，

其中还提供了一些重写方法例如 postInvalidate(int left,int top,int right,int bottom)来刷新一个矩形区域，以及延时执行。当然推荐的方法是通过一个 Handler 来处理的，可以在一个线程的 run 方法中调用 handler 对象的 postMessage 或 sendMessage 方法来实现，Android 程序内部维护着一个消息队列，会轮换处理这些消息。

1．Android Hander 类及其主要方法

Handler 类的主要作用主要有：在新启动的线程中发送消息，在主线程中获取和处理消息。Handler 类包含如下方法用于发送、处理消息。

① void handlerMessage(Message msg)：处理消息的方法，该方法通常用于被重写。

② final boolean hasMessage(int what)：检查消息队列中是否包含 what 属性为指定值的消息。

③ sendEmptyMessage(int what)：发送空消息。

④ final boolean sendMessage(Message msg)：立即发送消息。如果 Message 被成功放到 MessageQueue 里面则返回 true，反之，返回 false。

2．Handler 相关的组件

Hander 消息机制主要包括 4 个关键对象，分别是 Handler、Message、MessageQueue 和 Looper，Handler 消息处理流程简述如下：首先需要在 UI 线程中创建一个 Looper 对象，然后在子线程中调用 Handler 的 sendMessage()方法，接着这个消息会存在 UI 线程的 MessageQueue 中，通过 Looper 对象取出 MessageQueue 中的消息，最后发回 Handler 的 handleMessage()方法中。

下面对这 4 个对象进行简单说明。

（1）Handler

Handler 就是处理者的意思，消息的发送者和最终消息处理者，它主要用于发送消息和处理消息。一般使用 Handler 对象的 sendMessage 方法发送消息，发出的消息经过一系列的辗转处理后，最终会传递到 Handler 对象的 handlerMessage()方法中。

Handler 作为一个消息的管理者，其重要作用就是创建并发送消息，最后再处理消息。发送消息就是把指定的 Message 放入到消息队列中，等到合适的时机，消息泵从消息队列中抽取消息，再分发下去，进行处理。因此，在 Handler 中，有必要维护当前线程的 MessageQueue 和 Looper 的引用。对于一个线程来说，MessageQueue 和 Looper 都是唯一的，而多个 handler 是可以共享同一个线程的 MessageQueue 和 Looper 的引用。

（2）Message

Message 用于封装消息的简单数据结构，里面包含消息的 ID、数据对象、处理消息的 Handler 引用和 Runnable 等。一个 Message 本身就是一个消息队列，实际上是一个链式结构的类，它可以在内部携带少量的信息，用于在不同线程之间交换数据。Message 的 what 字段可以用来携带一些整型数据，obj 字段可以用来携带一个 Object 对象。

Android 中的 Handler 可以传递一些内容，通过 Bundle 对象可以封装 String、Integer 以及 Blob 二进制对象，我们通过在线程中使用 Handler 对象的 sendEmptyMessage 或 sendMessage 方法来传递一个 Bundle 对象到 Handler 处理器。对于 Handler 类提供了重写方法 handleMessage(Message msg)来判断，通过 msg.what 来区分每条信息。

（3）MessageQueue

MessageQueue 是消息队列的意思，提供消息的添加、删除、获取等操作来管理消息队列，主要用来存放通过 Handler 发送的消息。通过 Handler 发送的消息会存在 MessageQueue 中等待处理，每个线程中只会有一个 MessageQueue 对象。它采用先进先出的方式来管理 Message，程序在创建 Looper 对象时，会在它的构造器中创建 MessageQueue。

（4）Looper

Android 中每一个 Thread 都跟着一个 Looper，Looper 可以帮助 Thread 维护一个消息队列，Looper 是每个线程中 MessageQueue 的管家，用于建立消息循环并管理消息队列（MessageQueue），不停地从消息队列中抽取消息，分发下去并执行，周而复始，直到抽取到的消息是退出消息，Looper 结束，线程即将退出。调用 Looper 的 loop()方法后，就会进入一个无限循环中。然后，每当发现 MessageQueue 中存在一条消息，就会将它取出，并传递到 Handler 的 handlerMessage()方法中进行处理。此外，每个线程也只会有一个 Looper 对象，在主线程中创建 Handler 对象时，系统已经创建了 Looper 对象，所以不用手动创建 Looper 对象，而在子线程中的 Handler 对象，需要调用 Looper.loop()方法开启消息循环。

在 Android 中还提供了一种有别于线程的处理方式，就是 Task 以及 AsyncTask，从开源代码中可以看到是针对 Concurrent 的封装，开发人员可以方便地处理这些异步任务。

7.4 使用 HttpURLConnection 访问网络

在实际 Android 程序开发中，经常需要与服务器交互数据，也就是访问网络，此时就需要用到 HttpURLConnection 对象。HttpURLConnection 连接网络不需要设置 socket，HttpURLConnection 并不是底层的连接，而是在底层连接上的一个请求。这就是为什么 HttpURLConneciton 只是一个抽象类，自身不能被实例化的原因。HttpURLConnection 只能通过 URL.openConnection()方法创建具体的实例。虽然底层的网络连接可以被多个 HttpURLConnection 实例共享，但每一个 HttpURLConnection 实例只能发送一个请求。请求结束之后，应该调用 HttpURLConnection 实例的 InputStream 或 OutputStream 的 close()方法以释放请求的网络资源，不过这种方式对于持久化连接没用。对于持久化连接，得用 disconnect()方法关闭底层连接的 socket。

HttpURLConnection 是 Java 的标准类，继承自 URLConnection，可用于向指定网站发送 GET 请求或 POST 请求。它在 URLConnection 的基础上提供了如下便捷的方法：

① int getResponseCode()：获取服务器的响应代码。

② String getResponseMessage()：获取服务器的响应消息。

③ String getResponseMethod()：获取发送请求的方法。

④ void setRequestMethod(String method)：设置发送请求的方法。

在一般情况下，如果只是需要 Web 站点的某个简单页面提交请求并获取服务器响应，HttpURLConnection 完全可以胜任。但在绝大部分情况下，Web 站点的网页可能没这么简单，这些页面并不是通过一个简单的 URL 就可访问的，可能需要用户登录而且具有相应的权限才可访问该页面。

HttpURLConnection 连接 URL 的主要步骤如下：

（1）创建一个 URL 对象

```
URL url = new URL(http://www.baidu.com);
```

（2）利用 HttpURLConnection 对象从网络中获取网页数据

```
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
```

（3）设置连接超时

```
conn.setConnectTimeout(6*1000);
```

（4）对响应码进行判断

```
if (conn.getResponseCode() != 200)  //发送请求,将获取的网页以流的形式读回来
throw new RuntimeException("请求 url 失败");
```

（5）得到网络返回的输入流

```
InputStream is = conn.getInputStream();
```

（6）String result = readData(is, "GBK"); //文件流输入出文件用 outStream.write

（7）conn.disconnect();

上述步骤说明了手机终端与服务器建立连接并获取服务器返回数据的过程。需要注意的是，在使用对象访问网络时，需要设置超时限制时间，Android 系统在超过默认时间会收回资源、中断操作。如果不设置超时限制，在网络异常的情况下，会因取不到数据而一直等待，导致程序僵死不会继续往后执行。

7.5 使用 HttpClient 访问网络

为了更好地处理向 Web 站点请求，包括处理 Session、Cookie 等细节问题，Apache 开源组织提供了一个 HttpClient 项目，看它的名称就知道，它是一个简单的 HTTP 客户端（并不是浏览器），可以用于发送 HTTP 请求，接收 HTTP 响应。但不会缓存服务器的响应，不能执行 HTML 页面中嵌入的 JavaScript 代码；也不会对页面内容进行任何解析和处理。

简单来说，HttpClient 就是一个增强版的 HttpURLConnection，HttpURLConnection 可以做的事情 HttpClient 全部可以做；HttpURLConnection 没有提供的有些功能，HttpClient 也提供了，但它只是关注于如何发送请求、接收响应，以及管理 HTTP 连接。

Android 已经成功地集成了 HttpClient，这意味着开发人员可以直接在 Android 应用中使用 HttpClient 来访问提交请求、接收响应。例如一个 Android 应用程序需要向指定页面发送请求，但该页面并不是一个简单的页面，只有当用户已经登录，而且登录用户的用户名有效时才可访问该页面。如果使用 HttpURLConnection 来访问这个被保护的页面，那么需要处理的细节就太复杂了。其实访问 Web 应用中被保护的页面，使用浏览器则十分简单，用户通过系统提供的登录页面登录系统，浏览器会负责维护与服务器之间的 Session，如果用户登录的用户名、密码符合要求，就可以访问被保护资源了。在 Android 应用程序中，则可使用 HttpClient 来登录系统，只要应用程序使用同一个 HttpClient 发送请求，HttpClient 会自动维护与服务器

之间的 Session 状态，也就是说程序第一次使用 HttpClient 登录系统后，接下来使用 HttpClient 即可访问被保护页面了。

使用 HttpClient 发送请求、接收响应很简单，只要如下几步即可。

（1）创建 HttpClient 对象。

（2）创建请求方法的实例，并指定请求 URL。如果需要发送 GET 请求，创建 HttpGet 对象；如果需要发送 POST 请求，创建 HttpPost 对象。

（3）如果需要发送请求参数，可调用 HttpGet、HttpPost 共同的 setParams(HetpParams params)方法来添加请求参数；对于 HttpPost 对象而言，也可调用 setEntity(HttpEntity entity)方法来设置请求参数。

（4）调用 HttpClient 对象的 execute(HttpUriRequest request)发送请求，执行该方法返回一个 HttpResponse。

（5）调用 HttpResponse 的 getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头；调用 HttpResponse 的 getEntity()方法可获取 HttpEntity 对象，该对象包装了服务器的响应内容。程序可通过该对象获取服务器的响应内容。

（6）释放连接。无论执行方法是否成功，都必须释放连接。



【任务 7-1】 获取指定城市的天气预报



【任务描述】

编写程序通过网络请求服务器端的数据，获取并展示指定城市的天气预报。



【知识索引】

（1）OnClickListener 类、setOnClickListener()方法。

（2）WebView 类及 setJavaScriptEnabled()、setWebChrome()、setWebViewClient()、loadUrl()、setInitialScale()等方法。



【实施过程】

1．创建 Android 项目 App0701

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0701。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 1 个水平线性布局管理器 LinearLayout 和 1 个用于显示网页的 WebView 控件，接着在嵌套布局 LinearLayout 中添加 3 个 Button 控件，将 WebView 控件的 id 属性设置为 WebView1，将 Button 控件的 id 属性分别设置为 bj、sh、tj。

3．完善类 MainActivity 与实现程序功能

（1）定义类 MainActivity 与声明对象

类 MainActivity 的基本定义代码如下所示，该类继承自父类 Activity，并实现了接口 OnClickListener，用于添加单击事件监听器。

```
public class MainActivity extends Activity implements OnClickListener{
……
}
```

在 MainActivity 类定义中，首先声明 1 个 WebView 控件对象，代码如下所示。

```
private WebView webView;  //声明 WebView 控件的对象
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取布局中的所有对象，然后通过调用方法 setOnClickListener()为各个 Button 控件对象添加单击事件监听器。接着调用 WebView 控件对象的多个方法进行相关处理，首先设置该控件允许使用 JavaScript，以及处理 JavaScript 对话框和各种请求事件，再为该控件指定要加载的天气预报信息，最后将网页内容放大 3 倍，其代码如表 7-1 所示。

表 7-1　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	setTitle("天气查询");
06	Button bj=(Button)findViewById(R.id.bj);　　//获取布局管理器中添加的“北京”按钮
07	bj.setOnClickListener(this);
08	Button sh=(Button)findViewById(R.id.sh);　　//获取布局管理器中添加的“上海”按钮
09	sh.setOnClickListener(this);
10	Button gz=(Button)findViewById(R.id.tj);　　//获取布局管理器中添加的“天津”按钮
11	gz.setOnClickListener(this);
12	webView=(WebView)findViewById(R.id.webView1);　//获取 WebView 控件
13	webView.getSettings().setJavaScriptEnabled(true);　　//设置 JavaScript 可用
14	webView.setWebChromeClient(new WebChromeClient());　//处理 JavaScript 对话框
15	//处理各种通知和请求事件，如果不使用该句代码，将使用内置浏览器访问网页
16	webView.setWebViewClient(new WebViewClient());
17	webView.loadUrl("http://cache.xixik.com.cn/8/beijing/"); //设置默认显示的天气预报
18	webView.setInitialScale(57*3);　　//网页内容放大 3 倍
19	}

（3）重写方法 onClick()的代码实现程序功能

方法 onClick()的代码如表 7-2 所示，其功能是为屏幕中各个按钮的单击事件设置不同的响应，实现单击各个按钮时，传递不同的参数，调用自定义方法 openUrl()获取不同城市的天气预报信息。

表 7-2　方法 onClick()的代码

序号	代　　码
01	public void onClick(View view){

续表

序号	代　　码
02	switch(view.getId()){
03	case R.id.bj:　　//单击的是“北京”按钮
04	openUrl("beijing");
05	break;
06	case R.id.sh:　　//单击的是“上海”按钮
07	openUrl("shanghai");
08	break;
09	case R.id.tj:　　//单击的是“天津”按钮
10	openUrl("tianjin");
11	break;
12	}
13	}

自定义打开网页的方法 openUrl()的代码如表 7-3 所示，其功能是根据传入的参数不同，打开网页，获取并显示天气预报信息。

表 7-3　自定义方法 openUrl()的代码

序号	代　　码
01	private void openUrl(String id){
02	//获取并显示天气预报信息
03	webView.loadUrl("http://cache.xixik.com.cn/8/"+id+"/");
04	}

【说明】：天气预报网（网址为 http://cache.xixik.com.cn）中提供了免费的天气预报插件，使用该插件可以实现在 Android 应用程序中获取指定城市的天气预报。

4．在 AndroidManifest.xml 文件中增加允许访问网络资源的权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加允许访问网络资源的权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

5．程序运行与功能测试

Android 项目 App0701 的初始运行状态如图 7-2 所示，首先显示默认城市北京的天气预报信息。



图 7-2　Android 项目 App0701 的初始运行状态

单击左侧各个城市按钮，将显示对应城市的天气预报信息，这里，单击【上海】按钮，将显示上海的天气预报信息，单击【天津】按钮，将显示天津的天气预报信息。

【任务 7-2】实现百度在线搜索

【任务描述】

当用户在查找输入框中输入关键字然后单击按钮时，发送 GET 请求到百度服务器，然后百度服务器将搜索结果返回给手机客户端并展示，编写程序使用 HttpClient 对象的 GET 方式实现这一功能。

【知识索引】

（1）OnClickListener 类、setOnClickListener()方法、onClick()方法。

（2）Handler 类、handleMessage()方法、sendEmptyMessage()方法。

（3）Thread 类、run()方法、start()方法。

（4）ProgressBar 类、setVisibility()、BufferedReader 类、close()方法、URI 类。

（5）HttpClient 类、DefaultHttpClient 类。

（6）HttpGet 类、setURI()方法、HttpResponse 类、getStatusLine()方法、getStatusCode()方法、getEntity()方法。

（7）Log 类、e()方法。

【实施过程】

1．创建 Android 项目 App0702

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0702。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 1 个 EditText 控件、1 个 Button 控件、1 个 TextView 和 1 个 ProgressBar 控件，并设置好各个控件的必要属性。布局文件 activity_main.xml 中控件对应的代码如表 7-4 所示。

表 7-4　布局文件 activity_main.xml 中控件对应的代码

序号	布 局 代 码
01	<LinearLayout ……>
02	<!-- 定义输入搜索关键字的输入框 -->
03	<EditText
04	android:id="@+id/et"
05	……
06	android:text="北京" />
07	<!-- 定义提交 GET 请求的按钮 -->
08	<Button
09	android:id="@+id/btn"
10	……
11	android:text="百度搜索" />

续表

序号	布 局 代 码
12	<!-- 定义结果显示框控件 -->
13	<TextView
14	android:id="@+id/tv_show"
15	…… />
16	<!-- 定义进度条控件 -->
17	<ProgressBar
18	android:id="@+id/progressBar"
19	style="?android:attr/progressBarStyleLarge"
20	……
21	android:layout_centerHorizontal="true"
22	android:layout_centerVertical="true"
23	android:visibility="gone" />
24	</LinearLayout>

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如表 7-5 所示。

表 7-5　MainActivity 类中声明的多个对象

序号	代　　码
01	private TextView mTvShow;　　//用于显示搜索结果
02	private EditText mEtInPut;　　//输入文本框
03	private Button mBtn;　　// 提交按钮
04	private ProgressBar mProgressBar;　　// 进度框
05	private String mReturnConnection;
06	// 4.0 之后不允许在主线程中运行耗时的网络请求
07	private Handler mHandler = new Handler() {
08	@Override
09	public void handleMessage(Message msg) {
10	mProgressBar.setVisibility(View.GONE);
11	mTvShow.setText(mReturnConnection);
12	super.handleMessage(msg);
13	}
14	};

表 7-5 中第 07～14 行定义了一个 Handler 对象，用来接收其他线程给主线程发送的 message 消息。

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取布局中的所有对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 7-6 所示。

表 7-6　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 获取控件对象
06	mBtn = (Button) findViewById(R.id.btn);
07	mEtInPut = (EditText) findViewById(R.id.et);
08	mTvShow = (TextView) findViewById(R.id.tv_show);
09	mProgressBar = (ProgressBar) findViewById(R.id.progressBar);
10	// 绑定控件事件
11	setListener();
12	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()实现代码如表 7-7 所示。

表 7-7　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	// 添加事件
03	mBtn.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View arg0) {
06	mProgressBar.setVisibility(View.VISIBLE);
07	//开启线程进行 GET 请求
08	new Thread() {
09	@Override
10	public void run() {
11	// 发送 GET 请求
12	doGet();
13	//传递 handler 的消息
14	mHandler.sendEmptyMessage(0);
15	super.run();
16	}
17	}.start();
18	}
19	});
20	}

当用户单击【百度搜索】按钮时，执行表 7-7 中第 08～17 行代码，启动一个线程，发送 GET 请求，然后发送 message 消息给 handler 对象。

（4）自定义 doGet()方法

doGet()方法的代码如表 7-8 所示。

表 7-8　doGet()方法的代码

序号	代　　码
01	public void doGet() {
02	//定义接受的返回字节流对象
03	BufferedReader in = null;
04	String URL = "http://www.baidu.com/s?wd=" + mEtInPut.getText().toString();
05	try {
06	// http 客户端
07	HttpClient client = new DefaultHttpClient();
08	// 请求方式 GET
09	HttpGet request = new HttpGet();
10	// 设置请求地址
11	request.setURI(new URI(URL));
12	//定义相应对象
13	HttpResponse response = client.execute(request);
14	if (response.getStatusLine().getStatusCode() == 200) {
15	// 得到响应字符串
16	mReturnConnection = EntityUtils.toString(response.getEntity());
17	} else {
18	mReturnConnection = "Error Response: " + response.getStatusLine().toString();
19	}
20	} catch (Exception e) {
21	Log.e("app0702", e.toString());
22	} finally {
23	//关闭字节流对象
24	if (in != null) {
25	try {
26	in.close(); //关闭流对象
27	} catch (IOException ioe) {
28	Log.e("app0702", ioe.toString());
29	}
30	}
31	}
32	}

方法 doGet()中第 07 行定义了 HttpClient 客户端，第 09 行定义了 HttpGet 请求对象，第 11 行设置 HttpGet 的请求网址，第 13 行通过客户端 client 的 execute 方法执行对应的 HttpGet 请求，得到返回结果为 response，第 14 行通过 response 的状态判断是否请求成功，如果请求成功第 16 行即可得到网络返回的数据。

4．在 AndroidManifest.xml 文件中增加允许访问网络资源的权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加允许访问网络资源的权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

5．程序运行与功能测试

Android 项目 App0702 的初始运行状态如图 7-3 所示。

单击【百度搜索】按钮，显示的搜索结果如图 7-4 所示。这里显示的结果是 HTML 代码，其原因是没有对服务器返回的数据进行解析就予以显示，实际应用中要对网站返回的数据进行解析显示。



图 7-3　Android 项目 App0702 的初始运行状态

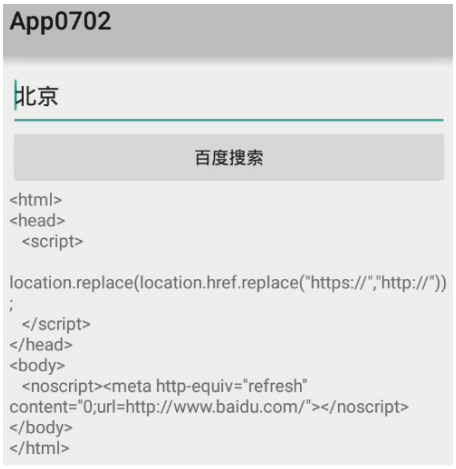


图 7-4　“百度搜索”结果

【任务 7-3】实现浏览网络图片

【任务描述】

编写程序使用 HttpURLConnection 对象的 GET 方式实现浏览网络图片的功能。

【知识索引】

（1）Handler 类、handleMessage()方法。

（2）Bitmap 类、ImageView 类、setImageBitmap()方法。

（3）Thread 类、run()方法、start()方法。

（4）HttpURLConnectio 类、setRequestMethod()、setConnectTimeout()、setRequestProperty()、getResponseCode()、getInputStream()等方法。

（5）URL 类、openConnection()方法。

（6）InputStream 类、Bitmap 类、BitmapFactory 类、decodeStream()方法。

（7）Message 类、what 属性、obj 属性。

【实施过程】

1．创建 Android 项目 App0703

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0703。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 1 个 EditText 控件、1 个 Button 控件和 1 个 ImageView 控件，将 EditText 控件的 id 属性设置为 et_path、hint 属性设置为“请输入网络图片路径”、text 属性设置为 http://www.baidu.com/img/bd_logo1.png，将 Button 控件的 text 属性设置为“浏览”，onClick 属性设置为“click”，将 ImageView 控件的 id 属性设置为 iv。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
protected static final int CHANGE_UI = 1;
protected static final int ERROR = 2;
private EditText etPath;
private ImageView iv;
```

（2）定义 Handler 对象

在 MainActivity 类定义中定义 Handler 对象，创建消息处理器，其代码如表 7-9 所示。

表 7-9　MainActivity 类中定义 Handler 对象的代码

序号	代　　码
01	// 主线程创建消息处理器
02	private Handler handler = new Handler(){
03	public void handleMessage(android.os.Message msg) {
04	if(msg.what == CHANGE_UI){
05	Bitmap bitmap = (Bitmap) msg.obj;
06	iv.setImageBitmap(bitmap);
07	}else if(msg.what == ERROR){
08	Toast.makeText(MainActivity.this, "显示图片错误",Toast.LENGTH_SHORT).show();
09	}
10	};
11	};

（3）在 onCreate()方法中编写代码实现程序功能

onCreate()方法的代码如表 7-10 所示，用于获取布局的 EditText 控件和 ImageView 控件。

表 7-10　onCreate()方法的代码

序号	代　　码
01	protected void onCreate(Bundle savedInstanceState) {
02	super.onCreate(savedInstanceState);
03	setContentView(R.layout.activity_main);
04	etPath = (EditText) findViewById(R.id.et_path);
05	iv = (ImageView) findViewById(R.id.iv);
06	}

（4）自定义方法 click

方法 click()的代码如表 7-11 所示，该方法中创建并开启了一个新线程，用于从网络上获取指定图片文件。首先创建一个 URL 对象，然后通过 URL 对象去获取 HttpURLConnection 对象，并且设置请求的方法、超时时间、请示头信息，最后获取服务器返回的输入流。

表 7-11　方法 click()的代码

序号	代　　码
01	public void click(View view) {
02	final String path = etPath.getText().toString().trim();
03	if (TextUtils.isEmpty(path)) {
04	Toast.makeText(this, "图片路径不能为空", Toast.LENGTH_SHORT).show();
05	} else {
06	//子线程请求网络,Android4.0 以后访问网络不能放在主线程中
07	new Thread() {　　//start()
08	public void run() {
09	// 连接服务器 get 请求 获取图片.
10	try {
11	URL url = new URL(path);　　//创建 URL 对象
12	// 根据 url 发送 http 的请求.
13	HttpURLConnection conn = (HttpURLConnection) url.openConnection();
14	// 设置请求的方式
15	conn.setRequestMethod("GET");
16	//设置超时时间
17	conn.setConnectTimeout(5000);
18	//设置请求头 User-Agent 浏览器的版本
19	conn.setRequestProperty(
20	"User-Agent",
21	"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; " +
22	"SV1; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; " +
23	".NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; Shuame)");
24	// 得到服务器返回的响应码
25	int code = conn.getResponseCode();
26	//请求网络成功后返回码是 200
27	if (code == 200) {
28	//获取输入流
29	InputStream is = conn.getInputStream();
30	//将流转换成 Bitmap 对象
31	Bitmap bitmap = BitmapFactory.decodeStream(is);
32	//iv.setImageBitmap(bitmap);
33	//TODO: 告诉主线程一个消息:帮我更改界面。内容:bitmap
34	Message msg = new Message();
35	msg.what = CHANGE_UI;
36	msg.obj = bitmap;

续表

序号	布 局 代 码
37	handler.sendMessage(msg);
38	} else {
39	//返回码不是 200　请求服务器失败
40	Message msg = new Message();
41	msg.what = ERROR;
42	handler.sendMessage(msg);
43	}
44	} catch (Exception e) {
45	e.printStackTrace();
46	Message msg = new Message();
47	msg.what = ERROR;
48	handler.sendMessage(msg);
49	}
50	};
51	}.start();
52	}
53	}

4．在 AndroidManifest.xml 文件中增加访问网络资源的权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加允许访问网络资源的权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
```

5．程序运行与功能测试

Android 项目 App0703 运行时，在文本框中输入一个网络图片的地址，例如 http://www.baidu.com/img/bd_logo1.png，然后单击【浏览】按钮，即可显示该图片，如图 7-5 所示。



图 7-5　Android 项目 App0703 运行时显示网络图片

【任务 7-4】实现短信发送

【任务描述】

编写 Android 程序实现短信发送功能。

【知识索引】

（1）OnClickListener 类、setOnClickListener()方法、onClick()方法。

（2）SmsManager 类、getDefault()方法、sendTextMessage()方法。

（3）PendingIntent 类、getBroadcast()方法。

【实施过程】

1．创建 Android 项目 App0704

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0704。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 2 个 EditText 和 1 个 Button 控件，将 EditText 控件的 id 属性分别设置为 etPhone 和 etMessage、hint 属性设置为“请输入接收短信的手机号码”和“测试发送短信功能”、text 属性设置为“18807319866”和“请输入短信内容”，将 Button 控件的 id 属性设置为 etMessage、text 属性设置为“发送短信”。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private EditText et1;      // 声明布局中的输入电话号码的控件
private EditText etMsg;    // 声明布局中的输入短信内容的控件
private Button btn1;       // 声明布局中的发送短信的 Button 控件
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取布局中的所有对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 7-12 所示。

表 7-12　onCreate()方法的代码

序号	代　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	// 得到布局中的开始加载的 EditText 的对象
06	et1 = (EditText) findViewById(R.id.etPhone);
07	// 得到布局中的开始加载的 EditText 的对象
08	etMsg = (EditText) findViewById(R.id.etMessage);
09	// 得到布局中的开始加载的 Button 的对象
10	btn1 = (Button) findViewById(R.id.btnSend);
11	//设置对象的监听器
12	setListener();
13	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()实现代码如表 7-13 所示。

表 7-13　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	// 设置 btn 的点击监听器
03	btn1.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View v) {
06	//得到用户需要发送的短信的内容
07	String msg = etMsg.getText().toString();
08	//得到用户需要接受短信的电话号码
09	String number = et1.getText().toString();
10	//当短信的接受电话号码和短信内容都不为空时进入分支
11	if (!"".equals(msg) && !"".equals(number)) {
12	//得到系统的 SmsManger 对象
13	SmsManager smsManager = SmsManager.getDefault();
14	//初始化 PendingIntent
15	PendingIntent sendIntent = PendingIntent.getBroadcast(
16	MainActivity.this, 0, new Intent(), 0);
17	//通过 smsManager 的 sendTextMessage 方法来发送短信
18	smsManager.sendTextMessage(number, null, msg, sendIntent, null);
19	//通过 Toast 提示短信发送成功
20	Toast.makeText(MainActivity.this, "发送成功",Toast.LENGTH_SHORT).show();
21	} else {
22	//通过 Toast 提示用户短信号码或者短信内容为空
23	Toast.makeText(MainActivity.this, "手机号码或短信内容为空...",
24	Toast.LENGTH_SHORT).show();
25	}
26	}
27	});
28	}

4．在 AndroidManifest.xml 文件中增加所需的权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加允许访问网络资源和发送短信的权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.SEND_SMS"/>
```

5．程序运行与功能测试

Android 项目 App070 的初始运行状态如图 7-6 所示，在电话号码输入框中输入接收短信的电话号码，在短信内容输入框输入短信内容，然后单击【发送短信】按钮，如果短信发送成

功，则会显示如图 7-7 所示的提示信息。



图 7-6　Android 项目 App070 的初始运行状态



图 7-7　“发送成功”的提示信息

【任务 7-5】实现电话拨打

【任务描述】

编写 Android 程序通过单击按钮实现电话拨打功能。

【知识索引】

（1）OnClickListener 类、setOnClickListener()方法、onClick()方法。

（2）Uri 类、parse 方法()。

（3）Intent 类、ACTION_CALL 常量。

（4）Activity 类、startActivity 方法()。

【实施过程】

1．创建 Android 项目 App0705

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0705。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），且将默认添加的 TextView 控件删除，然后添加 1 个 EditText 控件和 1 个 Button 控件，将 EditText 控件的 id 属性设置为 Et、hint 属性设置为“请输入拨打的电话号码”、text 属性设置为“17766668888”，将 Button 控件的 id 属性设置为 bt、text 属性设置为“单击开始拨打电话”。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private Button btn;    //声明布局中的开始跳转 Button 控件
private EditText et;    //声明布局中的输入电话号码控件
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取布局中的所有对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 7-14 所示。

表 7-14　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {

续表

序号	代　　码
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	//得到布局中的开始加载的 Button 的对象
06	btn = (Button)findViewById(R.id.Btn);
07	//得到布局中的开始加载的 EditText 的对象
08	et = (EditText)findViewById(R.id.Et);
09	// 设置对象的监听器
10	setListener();
11	}

（3）编写代码为按钮自定义单击监听器

编写代码定义单击事件监听器 OnClickListener，方法 setListener()实现代码如表 7-15 所示。

表 7-15　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	// 设置按钮的单击事件监听器
02	private void setListener() {
03	//设置 btn 的点击监听器
04	btn.setOnClickListener(new OnClickListener() {
05	@Override
06	public void onClick(View v) {
07	//通过 EditView 的 gettext 方法得到用户输入的电话号码
08	String phonenum = et.getText().toString();
09	//如果电话号码不为空的话
10	if (!"".equals(phonenum)) {
11	//通过电话号码构成 uri 对象，需要 tel:加上电话号码
12	Uri uri = Uri.parse("tel:" + phonenum);
13	//通过 Intent.ACTION_CALL 打开上述 uri
14	Intent it = new Intent(Intent.ACTION_CALL, uri);
15	//通过 intent 打开 activity
16	startActivity(it);
17	}
18	else{
19	//如果用户输入的电话号码为空，通过 Toast 提示用户输入
20	Toast.makeText(MainActivity.this,"请输入您要拨打的电话号码...",
21	Toast.LENGTH_SHORT).show();
22	}
23	}
24	});
25	}

表 7-15 中第 08 行获取用户输入的电话号码，如果为空第 20 行代码通过 Toast 提示用户再次输入，如果不为空第 12 行代码通过用户输入的电话号码初始化 Uri 对象，然后第 14 行代码定义 Intent 对象，并设置 Action 为 Intent.ACTION_CALL，然后显示对象为 uri，最后第 16 行代码调用 startActivity()方法打开对应的 Activity，这样就可以显示拨号界面直接拨打电话了。

4．在 AndroidManifest.xml 文件中增加所需的权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加允许访问网络资源和拨打电话的权限。

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.SEND_SMS"/>
```

5．程序运行与功能测试

Android 项目 App0705 的初始运行状态如图 7-8 所示。在电话号码输入框中输入拨打的电话号码，然后单击【单击开始拨打电话】按钮，开始拨打电话，且显示如图 7-9 所示的电话拨打界面。



图 7-8　Android 项目 App0705 的初始运行状态

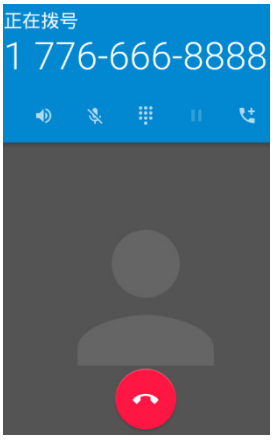


图 7-9　电话拨打时显示的拨打界面



Hander 消息机制主要包括 4 个关键对象，分别是 Handler、Message、MessageQueue 和 Looper。HttpURLConnection 是 Java 的标准类，继承自 URLConnection，可用于向指定网站发送 GET 请求或 POST 请求。HttpClient 就是一个增强版的 HttpURLConnection，Android 已经成功地集成了 HttpClient，这意味着开发人员可以直接在 Android 应用中使用 Httpclient 来访问提交请求、接收响应。

本单元主要阐述了 HTTP 协议、URL 请示的类别、Android 的线程与 Hander 消息机制，通过多个实例分析和编程学会使用 HttpURLConnection 访问网络、使用 HttpClient 访问网络等方法，以及短信发送和电话拨打等通信应用的实现。



单元习题

一、填空题

1．Android 系统提供了多种网络通信方式，包括（　　）、（　　）、（　　）和 WebView。

2．当客户端与服务器端建立连接后，向服务器发送的请求，被称为（　　）。

3. Android 客户端访问网络发送 HTTP 请求的方式一般有两种：（　　）和（　　）。

4．与服务器交互过程中，最常用的两种数据提交方式是（　　）和（　　）。

5. 为了根据下载进度实时更新 UI 界面，需要用到 Handler 消息机制来实现（　　）。

6．HttpURLConnection 是一个标准的（　　）类。

二、选择题

1．下列通信方式中，不是 Android 系统提供的是（　　）。

A．Socket 通信　　B．HTTP 通信　　C．URL 通信　　D．以太网通信

2．关于 HttpURLConnection 访问网络的基本用法，描述错误的是（　　）。

A．HttpURLConnection 对象需要设置请求网络的方式

B．HttpURLConnection 对象需要设置超时时间

C．需要通过 new 关键字来创建 HttpURLConnection 对象

D．访问网络完毕需要关闭 HTTP 链接

3．下列选项中，不属于 AsyncHttpClient 特点的是（　　）。

A．发送异步 HTTP 请求

B．HTTP 请求发生在 UI 线程之外

C．内部采用了线程池来处理并发请求

D．自动垃圾回收

4．下列选项中，关于 GET 和 POST 请求方式，描述错误的是（　　）。

A．使用 GET 方式访问网络 URL 的长度是有限制的

B．HTTP 协议规定 GET 方式请求 URL 的长度不超过 2K

C．POST 方式对 URL 的长度是没有限制的

D．GET 请求方式向服务器提交的参数跟在请求 URL 后面

5．Hanlder 是线程与 Activity 通信的桥梁，如果线程处理不当，机器就会变得越慢，线程销毁的方法是(　　)。

A．onDestroy()　　B．onClear()　　C．onFinish()　　D．onStop()

6．下列选项中，不属于 Handler 机制中的关键对象是（　　）。

A．Content　　B．Handler　　C．MessageQueue　　D．Looper

三、简答题

1．简要说明 GET 请求与 POST 请求的主要区别。

2．简述 Handler 机制四个关键对象的作用。

3．简述使用 HttpURLConnection 访问网络的基本步骤。

4．简述使用 HttpClient 访问网络的基本步骤。

单元 8

Android 的图像浏览与图形绘制程序设计

Android 提供了 ImageView 控件显示普通静态图片，也提供了 AnimationDrawable 类来开发逐帧动画，还可通过 Animation 对普通图片使用补间动画。图形、图像处理不仅对 Android 系统的应用界面非常重要，而且 Android 系统上益智类游戏、2D 游戏都需要大量的图形、图像处理。

在 Android 中绘图时，常用到的几个类是 Bitmap、BitmaptFactory、Paint、Canvas 和 Color，这些类都存放在 android.graphics 包中。其中，Paint 类代表画笔，Canvas 类代表画布，Color 代表颜料，有了 Paint 和 Canvas 类就可以进行绘图操作了。



教学导航

教学目标	（1）熟悉 Bitmap、BitmapFactory 类的功能及其常用方法 （2）熟悉 Paint、Canvas、Color、Path 等类的功能、主要方法及其典型应用 （3）学会应用图形图像类设计图片浏览器 （4）学会应用图形图像类绘制几何图形和多种形式的路径
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	6 课时



知识导读

8.1 使用简单图片

图片不仅可以使用 ImageView 控件来显示，也可以作为 Button、window 的背景。从广义角度看，Android 应用中的图片不仅包括*.png、*.jpg、*.gif 等各种格式的位图，也包括使用 XML 资源文件定义的各种 Drawable 资源。

1．使用 Drawable 资源

当我们为 Android 应用增加了 Drawable 资源后，Android SDK 会为这份资源在 R 清单文件中创建一个索引项：R.drawable.file_name。接下来可以在 XML 资源文件中通过 @drawable/file_name 来访问该 Drawable 对象，也可在 Java 代码中通过 R.drawable.file_name 访问该 Drawable 对象。

需要指出的是，R.drawable.file_name 是一个 int 类型的常量，它只代表 Drawable 对象的 ID，如果 Java 程序中需要获取实际的 Drawable 对象，则可以调用 Resources 的 getDrawable（int id）方法获取。

2．Bitmap 类

Bitmap 代表一张位图，BitmapDrawable 里封装的图片就是一个 Bitmap 对象。开发者为了把一个 Bitmap 对象包装成 BitmapDrawable 对象，可调用 BitmapDrawable 构造器，代码如下：

```
BitmapDrawable drawable=new BitmapDrawable(bitmap);
```

如果需要获取 BitmapDrawable 所包装的 Bitmap 对象，则可以调用 BitmapDrawable 的 getBitmap()方法，代码如下：Bitmap bitmap=drawable.getBitmap（）;

Bitmap 还提供了一些静态方法来创建新的 Bitmap 对象，常用静态方法如下：

① createBitmap(Bitmap source，int x，int y，int width，int height)：从源位图 source 的指定坐标(x、y)开始，从中“挖取”宽 width、高 height 的一块出来，创建新的 Bitmap 对象。

② createScaleBitmap（Bitmap src，int dstWidth，int dstHeight，boolean filter）：对源位图 src 进行缩放，缩放成宽 dstWidth、高 dstHeight 的新位图。

③ createBitmap（int width，int height，Bitmap.Config config）：创建一个宽 width、高 height 的新位图。

④ createBitmap（Bitmap source，int x，int y，int width，int height，Matrix m，boolean filter）：从源位图 source 的指定坐标(x，y)开始，从中挖取宽 width、高 height 的一块出来创建新的 Bitmap 对象，并按 Matrix 指定的规则进行变换。

Bitmap 类常用的方法如下：

① public void recycle()：获取位图占用的内存空间。

② public final boolean isRecycled()：判断位图内存是否已释放。

③ public final int getWidth()：获取位图的宽度。

④ public final int getHeight()：获取位图的高度。

⑤ public final boolean isMutable()：判断图片是否可修改。

⑥ public int getScaledWidth(Canvas canvas)：获取指定密度转换后的图像宽度。

⑦ public int getScaledHeight(Canvas canvas)：获取指定密度转换后的图像高度。

⑧ public boolean compress(CompressFormat format, int quality, OutputStream stream)：按指定的图片格式以及画质，将图片转换为输出流。

3．BitmapFactory 类

BitmapFactory 是一个工具类，它用于提供大量的方法，这些方法可用于从不同的数据源来解析、创建 Bitmap 对象，BitmapFactory 类包含如下方法：

① decodeByteArray（byte[] data，int offset，int length）：从指定字节数组的 offset 位置开始，将长度为 length 的字节数据解析成 Bitmap 对象。

② decodeFile（String pathName）：从 pathName 指定的文件中解析、创建 Bitmap 对象。

③ decodeFileDescriptor（FileDescriptor fd）：用于从 FileDescriptor 对应的文件中解析、创建 Bitmap 对象。

④ decodeResource（Resources res，int id）：用于根据给定的资源 ID 从指定资源中解析、创建 Bitmap 对象

⑤ decodeStream（InputStream is）：用于从指定输入流中解析、创建 Bitmap 对象。

手机系统的内存比较小，如果系统不停地去解析、创建 Bitmap 对象，可能由于前面创建 Bitmap 所占用的内存还没有回收，从而导致程序运行时引发 OutOfMemory 错误。Android 为 Bitmap 提供了两个方法来判断它是否已回收，以及强制 Bitmap 回收自己：

① boolean isRecycled（）：返回该 Bitmap 是否已被回收。

② void recycle（）：强制一个 Bitmap 对象立即回收自己。

除此之外，如果 Android 应用需要访问其他存储路径（例如 SD 卡中）里的图片，都需要借助于 BitmapFactory 来解析、创建 Bitmap 对象。

8.2 位图的典型应用

1．存取位图

把 Bitmap 保存在 SDcard 卡中的代码如下：

```
File fImage = new File("/sdcard/dcim","ic_call_log_list_incoming_call.jpeg");
FileOutputStream iStream = new FileOutputStream(fImage);
```

取出 Bitmap 的代码如下：

```
oriBmp.compress(CompressFormat.JPEG, 100, iStream);
```

2．从资源中获取位图

可以使用 BitmapDrawable 或者 BitmapFactory 来获取资源中的位图。

首先需要获取资源：

```
Resources res=getResources();
```

（1）使用 BitmapDrawable 获取位图

① 使用 BitmapDrawable(InputStream is)构造一个 BitmapDrawable。

② 使用 BitmapDrawable 类的 getBitmap()获取得到位图。

```
InputStream is=res.openRawResource(R.drawable.pic180);
BitmapDrawable bmpDraw=new BitmapDrawable(is);
Bitmap bmp=bmpDraw.getBitmap();
```

或者采用下面的方式：

```
BitmapDrawable bmpDraw=(BitmapDrawable)res.getDrawable(R.drawable.pic180);
Bitmap bmp=bmpDraw.getBitmap();
```

（2）使用 BitmapFactory 获取位图

使用 BitmapFactory 类 decodeStream(InputStream is)方法解码位图资源，获取位图。

```
Bitmap bmp=BitmapFactory.decodeResource(res, R.drawable.pic180);
```

BitmapFactory 的所有函数都是 static，这个辅助类可以通过资源 ID、路径、文件、数据流等方式来获取位图。

以上方法在编程的时候可以自由选择，在 Android SDK 中说明可以支持的图片格式如下：png (preferred), jpg (acceptable), gif (discouraged)，和 bmp（Android SDK Support Media Format）。

3．显示位图

显示位图可以使用核心类 Canvas，通过 Canvas 类的 drawBirmap()方法显示位图，或者借助于 BitmapDrawable 来将 Bitmap 绘制到 Canvas。当然，也可以通过 BitmapDrawable 将位图显示到 View 中。

（1）转换为 BitmapDrawable 对象显示位图

```
Bitmap bmp=BitmapFactory.decodeResource(res, R.drawable.pic180);  //获取位图
BitmapDrawable bmpDraw=new BitmapDrawable(bmp);  //转换为 BitmapDrawable 对象
ImageView iv = (ImageView)findViewById(R.id.ImageView);  //显示位图
iv.setImageDrawable(bmpDraw);
```

（2）使用 Canvas 类显示位图

```
Bitmap bmp = BitmapFactory.decodeResource(getResources(), R.drawable.pic180);
canvas.drawColor(Color.BLACK);
canvas.drawBitmap(bmp, 10, 10, null);
```

8.3 绘图

除了使用已有的图片之外，Android 应用常常需要在运行时动态生成图片，例如一个手机游戏，游戏界面看上去丰富多彩，而且可以随着用户动作而动态改变，这就需要借助于 Android 的绘图支持了。

Android 2D Graphics 的绝大部分 API 都在 android.graphics 中，它提供了低级的 graphics 工具，包括 canvas，color filters，point，rectangle 等，我们可以使用它们直接在屏幕上绘制想要的图形。在 android.graphics 中还有一个子包，android.graphics.drawable，它定义了一系列的 drawable 对象。而这个包里还有一个子包，android.graphics.drawable.shapes，它定义了 ShapeDrawable 所使用的一系列 shape 对象。要掌握 Android 2D Graphics 必须要熟悉这三个包的各种 API。

1．Canvas（画布）类

Android 的绘图应该继承 View 组件，并重写它的 onDraw（Canvas canvas）方法即可。重写 onDraw（Canvas canvas）方法时涉及一个绘图类 Canvas，Canvas 代表了“依附”于指定 View 的画布，它提供了一些方法绘制各种图形。

Canvas 是画布的意思，表现在屏幕上就是一块区域，我们可以在上面使用各种 API 绘制我们想要的东西。可以说，Canvas 贯穿整个 2D Graphics，android.graphics 中的所有类，几乎都与 Canvas 有直接或间接的联系。所以了解 Canvas 是学习 2D Graphics 的基础。

（1）如何获得一个 Canvas 对象

Canvas 对象的获取方式有三种：

第一种我们通过重写 View.onDraw 方法，View 中的 Canvas 对象会被当做参数传递过来，我们操作这个 Canvas，效果会直接反应在 View 中。

第二种就是自己创建一个 Canvas 对象，一个 Canvas 对象一定是结合了一个 Bitmap 对象的，为一个 Canvas 对象设置一个 Bitmap 对象。

示例代码如下：

```
Bitmap b = Bitmap.createBitmap(100,100, Bitmap.Config.ARGB_8888);
Canvas c = new Canvas(b);
```

第三种方式，是调用 SurfaceHolder.lockCanvas()，返回一个 Canvas 对象。

（2）Canvas 的主要功能

Canvas 类提供了一系列的 draw 方法，从这些方法的名字就可以知道 Canvas 可以绘制的对象。

① 填充。填充的示例代码如下所示：

```
public void drawARGB(int a, int r, int g, int b)
public void drawColor(int color)
public void drawRGB(int r, int g, int b)
public void drawColor(int color, PorterDuff.Mode mode)
```

② 绘制几何图像。Canvas 类可以用来实现各种图形的绘制工作，如绘制直线、矩形、圆等。Canvas 绘制常用图形的方法如下：

canvas.drawArc：（绘制扇形）、canvas.drawCircle（绘制圆形）、canvas.drawOval（绘制椭圆）、canvas.drawLine（绘制直线）、canvas.drawPoint（绘制点）、canvas.drawRect（绘制矩形）、canvas.drawRoundRect（圆角矩形）、canvas.drawVertices（绘制顶点）、cnavas.drawPath（绘制路径）

③ 绘制图片。canvas.drawBitmap（绘制位图）、canvas.drawPicture（绘制图片）。

④ 绘制文本。canvas.drawText。

上面列举的是 Canvas 所能绘制的基本内容，在实际使用中，可以使用各种过滤或者过渡模式，或者其他手段，来达到绘制各种效果。

（3）Canvas 的变换

如果只有那些简单的 draw 方法，那么 canvas 的功能就太单调了。Canvas 还提供了如下方法进行变换：

① rotate（float degrees，float px，float py）：对 Canvas 执行旋转。

② scale（float sx，float sy，float px，float py）：对 Canvas 执行缩放。

③ translate（float dx，float dy）：移动 Canvas，向右移动 dx 距离（dx 为负时向左移），向下移动 dy 距离（dy 为负时向上移动）。

示例代码如下所示：

```
@Override
protected void onDraw(Canvas canvas) {
   canvas.translate(100, 100);
   canvas.drawColor(Color.RED);     //可整个屏幕依然填充为红色
   canvas.drawRect(new Rect(-100, -100, 0, 0), new Paint());  //缩放了
   canvas.scale(0.5f, 0.5f);
   canvas.drawRect(new Rect(0, 0, 100, 100), new Paint());
   canvas.translate(200, 0);
```

```
        canvas.rotate(30);
        canvas.drawRect(new Rect(0, 0, 100, 100), new Paint());   //旋转了
        canvas.translate(200, 0);
        canvas.skew(.5f, .5f);    //扭曲了
        canvas.drawRect(new Rect(0, 0, 100, 100), new Paint());
    }
```

Canvas 虽然内部保持了一个 Bitmap，但是它本身并不代表那个 Bitmap，而更像是一个图层。我们对这个图层的平移旋转和缩放等操作，并不影响内部的 Bitmap，仅仅是改变了该图层相对于内部 Bitmap 的坐标位置、比例和方向而已。

（4）Canvas 的保存和回滚

为了方便一些转换操作，Canvas 还提供了保存和回滚的方法(save 和 restore)，例如可以先保存目前画纸的位置(save)，然后旋转 90 度，向下移动 100 像素后画一些图形，画完后调用 restore 方法返回到刚才保存的位置。

2．Color（颜色）类

Android 系统中颜色的常用表示方法有以下 3 种：

（1）int color = Color.BLUE;

（2）int color = Color.argb(150,200,0,100);

（3）在 XML 文件中定义颜色。

在实际应用当中，常用的颜色有以下一些，其颜色常量及其表示的颜色如下所示：

Color.BLACK（黑色）、Color.GREEN（绿色）、Color.BLUE（蓝色）、Color.LTGRAY（浅灰色）、Color.CYAN（青绿色）、Color.MAGENTA（红紫色）、Color.DKGRAY（灰黑色）、Color.RED（红色）、Color.YELLOW（黄色）、Color.TRANSPARENT（透明）、Color.GRAY（灰色）Color.WHITE（白色）。

3．Paint（画笔）类

Canvas 提供的方法还涉及一个 API：Paint，Paint 代表了 Canvas 上的画笔，因此 Paint 类主要用于设置绘制风格，包括画笔颜色、画笔笔触粗细、填充风格等。

要绘制图形，首先要调整画笔，按照自己的开发需要设置画笔的相关属性。Pain 类的常用属性设置方法如下：

① setAntiAlias()：设置画笔的锯齿效果。

② setColor()：设置画笔的颜色。

③ setARGB()：设置画笔的 A、R、G、B 值。

④ setAlpha()：设置画笔的 Alpha 值，范围为 0～255。

⑤ setTextSize()：设置字体的尺寸。

⑥ setStyle()：设置画笔的风格（空心或实心）。

⑦ setStrokeWidth()：设置空心边框的宽度。

⑧ getColor()：获取画笔的颜色。

在 Canvas 提供的绘制方法中还用到了一个 API：Path，Path 代表任意多条直线连接而成的任意图形，当 Canvas 根据 Path 绘制时，它可以绘制出任意的形状。

任务实战

【任务 8-1】设计按钮切换的图片浏览器

【任务描述】

使用 ImageSwitcher 控件创建图片浏览器，实现类似于 Windows 操作系统中“Windows 照片查看器”中的上一张、下一张切换图片的功能。其运行结果如图 8-1 所示。

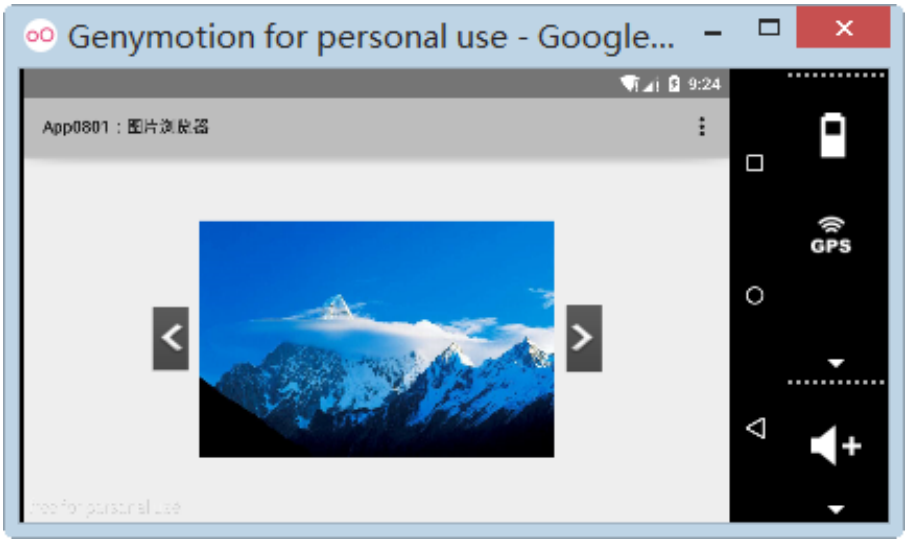


图 8-1　按钮切换图片浏览器的运行结果

【知识索引】

（1）ImageSwitcher 类及 setInAnimation()、setOutAnimation()、setFactory()、setImageResource()等方法。

（2）ImageView 类、setScaleType()方法、setLayoutParams()方法、FIT_CENTER 常量。

（3）AnimationUtils 类、loadAnimation()方法。

（4）LayoutParams 类、WRAP_CONTENT 常量。

（5）OnClickListener 类、setOnClickListener()方法、onClick()方法。

（6）ViewFactory 接口。

【实施过程】

1．创建 Android 项目 App0801 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0801，将本任务所需的图片文件导入或复制到 res/drawable 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

修改项目 App0801 的 res/layout 文件夹下的布局文件 activity_main.xml，将布局方式设置为水平线性布局，并将默认添加的 TextView 控件删除，然后添加两个 Button 控件和 1 个图片

切换器控件 ImageSwitcher，将 Button 控件的 id 属性分别设置为 btnFront 和 btnNext，将 ImageSwitcher 控件的 id 属性设置为 imageSwitcher。

3．完善类 MainActivity 与实现程序功能

（1）声明变量和数组

在主活动 MainActivity 中，首先声明并初始化一个保存待显示图片 id 的数组，然后声明一个保存当前显示图片索引的变量，最后声明一个图片切换器的对象。具体代码如下所示：

```
private int[] imageId = new int[] { R.drawable.t01, R.drawable.t02,
          R.drawable.t03, R.drawable.t04, R.drawable.t05,
          R.drawable.t06, R.drawable.t07, R.drawable.t08,
          R.drawable.t09, R.drawable.t10, };  // 定义并初始化保存图片 id 的数组
private int index = 0;                        // 当前显示图像的索引
private ImageSwitcher imgSwitcher;            // 声明一个图像切换器对象
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中，首先获取布局文件中添加的图像切换器，并为其设置淡入淡出的动画效果，然后调用其父类 ViewSwitcher 的 setFactory()方法指定视图切换工厂，其参数为 ViewSwitcher.ViewFactory 类型的对象，并重写 makeView()方法，最后为图片切换器设置默认显示的图片，主要代码如表 8-1 所示。

表 8-1　类 MainActivity 中 onCreate()方法的部分代码

序号	代　　码
01	imgSwitcher = (ImageSwitcher) findViewById(R.id.imageSwitcher); // 获取图像切换器
02	// 设置淡入动画效果
03	imgSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_in));
04	// 设置淡出动画效果
05	imgSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_out));
06	imgSwitcher.setFactory(new ViewFactory() {
07	@Override
08	public View makeView() {
09	// 实例化一个 ImageView 类的对象
10	ImageView imageView = new ImageView(MainActivity.this);
11	// 设置保持纵横比居中缩放图像
12	imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
13	imageView.setLayoutParams(new ImageSwitcher.LayoutParams(
14	LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
15	return imageView;　　// 返回 imageView 对象
16	}
17	});
18	imgSwitcher.setImageResource(imageId[index]);　　// 显示默认的图片

表 8-1 中第 03 行和第 05 行分别使用 ImageSwitcher 类的父类 ViewAnimator 的 setInAnimation()方法和 setOutAnimation()方法为图片切换器设置动画效果。

（3）编写代码为按钮添加单击事件监听器

编写代码获取用于控制显示图片的“上一张”和“下一张”按钮，并分别为其添加单击事件监听器，在重写的 onClick()方法中改变图片切换器中显示的图片，实现代码如表 8-2 所示。

表 8-2　类 MainActivity 中实现按钮单击事件监听器的代码

序号	代　　码
01	Button front = (Button) findViewById(R.id.btnFront);　// 获取“上一张”按钮
02	Button next = (Button) findViewById(R.id.btnNext);　// 获取“下一张”按钮
03	front.setOnClickListener(new OnClickListener() {
04	@Override
05	public void onClick(View v) {
06	if (index > 0) {
07	index--;
08	} else {
09	index = imageId.length - 1;
10	}
11	imgSwitcher.setImageResource(imageId[index]);　// 显示当前图片
12	}
13	});
14	next.setOnClickListener(new OnClickListener() {
15	@Override
16	public void onClick(View v) {
17	if (index < imageId.length - 1) {
18	index++;
19	} else {
20	index = 0;
21	}
22	imgSwitcher.setImageResource(imageId[index]);　// 显示当前图片
23	}
24	});

4．程序运行与功能测试

Android 项目 App0801 的运行结果如图 8-1 所示，单击【上一张】按钮显示前一张图片，单击【下一张】按钮显示后一张图片。

【任务 8-2】设计滑动切换的图片浏览器

【任务描述】

应用 ImageView 控件设计一款图片浏览器，实现通过单指在屏幕滑动切换图片的效果。其运行结果如图 8-2 所示。



图 8-2　滑动切换图片浏览器的运行结果

【知识索引】

（1）GestureDetector 类、OnGestureListener 类、SimpleOnGestureListener 类、MotionEvent 类、onTouchEvent()方法。

（2）ImageView 类、setImageResource()方法。

【实施过程】

1．创建 Android 项目 App0802 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0802，将本任务所需的图片文件导入或复制到 res/drawable 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

修改项目 App0802 的 res/layout 文件夹下的布局文件 activity_main.xml，将布局方式设置为水平线性布局，并将默认添加的 TextView 控件删除，然后添加 1 个 ImageView 控件，将 ImageView 控件的 id 属性设置为 imageView、src 属性设置为@drawable/t01。

3．完善类 MainActivity 与实现程序功能

（1）声明变量和数组

在主活动 MainActivity 中，首先声明并初始化一个保存待显示图片 id 的数组，然后分别声明一个保存当前显示图片序号的变量、手势监听器对象和保存 ImageView 的对象。具体代码如下所示。

```
private int[] resId = new int[]{
       R.drawable.t01, R.drawable.t02, R.drawable.t03,
       R.drawable.t04, R.drawable.t05, R.drawable.t06
};                                          //定义图片的资源数组
private int count = 0;                      //定义当前显示图片的序号
private GestureDetector gestureDetector;    //定义手势监听器对象
private ImageView iv;                       // 定义保存 ImageView 的对象
```

（2）在 onCreate()方法中编写代码实现程序功能

在类 MainActivity 的 onCreate()方法中，首先获取布局文件中添加的 ImageView 控件，然后设置手势监听器 gestureDetector 的处理效果由 onGestureListener 对象来处理，主要代码如

表 8-3 所示。

表 8-3　类 MainActivity 中 onCreate()方法的代码

序号	代　　码
01	protected void onCreate(Bundle savedInstanceState) {
02	super.onCreate(savedInstanceState);
03	setContentView(R.layout.activity_main);
04	//得到当前页面的 ImageView 控件
05	iv = (ImageView) findViewById(R.id.imageView);
06	//设置手势监听器的处理效果由 onGestureListener 来处理
07	gestureDetector = new GestureDetector(MainActivity.this,onGestureListener);
08	}

（3）编写代码自定义手势识别器实现图片滑动

编写代码实现当前 Activity 的 onTouchEvent 方法，然后将所有的 Touch 事件转交给手势监听器对象 gestureDetector 进行处理，实现代码如表 8-4 中第 02 行至第 05 行所示。

自定义 GestureDetector 的手势识别监听器，其中实现 onFinger 方法，当有滑动事件触发时自动回调此方法，在此回调方法中得到滑动两点之间的位置，然后通过两点位置的差别，判断此滑动是向左滑动，还是向右滑动，然后进行图片的切换，实现代码如表 8-4 中第 07 行至第 28 行所示。

表 8-4　类 MainActivity 中 onTouchEvent()方法和自定义手势识别器的代码

序号	代　　码
01	@Override
02	public boolean onTouchEvent(MotionEvent event) {
03	//当前 Activity 被触摸时回调
04	return gestureDetector.onTouchEvent(event);
05	}
06	//自定义了 GestureDetector 的手势识别监听器
07	private GestureDetector.OnGestureListener onGestureListener
08	= new GestureDetector.SimpleOnGestureListener() {
09	//当识别的手势是滑动手势时回调 onFinger 方法
10	@Override
11	public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
12	float velocityY) {
13	//得到滑动手势的起始和结束点的 x，y 坐标，并进行计算
14	float x = e2.getX() - e1.getX();
15	float y = e2.getY() - e1.getY();
16	//通过计算结果判断用户是向左滑动或者向右滑动
17	if (x > 0) {
18	count++;
19	count %= 3;
20	} else if (x < 0) {

续表

序号	代　　码
21	count--;
22	count = (count + 3) % 3;
23	}
24	//切换 imageview 的图片
25	iv.setImageResource(resId[count]);
26	return true;
27	}
28	};

4．程序运行与功能测试

Android 项目 App0802 的运行结果如图 8-2 所示，鼠标指针向左滑动显示前一张图片，鼠标指针向右滑动显示后一张图片。

【任务 8-3】绘制简单几何图形

【任务描述】

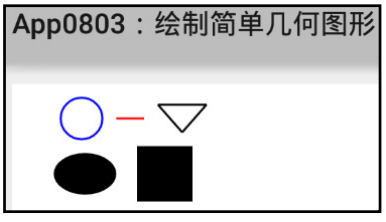


图 8-3　绘制几何图形的运行结果

在 Android 中，Canvas 类提供了丰富的绘制几何图形的方法，编写程序绘制圆形、直线、折线、椭圆和矩形等几何图形，其运行结果如图 8-3 所示。

【知识索引】

（1）FrameLayout 类、RectF 类。

（2）Paint 类及 setAntiAlias()、setStrokeWidth()、setStyle()、setColor()等方法。

（3）Canvas 类及 drawCircle()、setColor()、setStyle()、drawLine()、drawOval()、drawRect()等方法。

【实施过程】

1．创建 Android 项目 App0803

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0803。

2．完善布局文件 activity_main.xml 与界面设计

修改项目 App0803 的 res/layout 文件夹下的布局文件 activity_main.xml，将默认添加的 TextView 控件删除，然后将页面的相对布局修改为帧布局类型，用于显示自定义的绘图类。

3．完善类 MainActivity 与实现程序功能

（1）创建一个继承自 android.view.View 类的内部类

在类 MainActivity 中创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法 MyView()和重写 onDraw(Canvas canvas)方法。主体代码如表 8-5 所示。

表 8-5　内部类 MyView 的代码

序号	代　　码
01	public class MyView extends View{

续表

序号	代　　码
02	public MyView(Context context) {
03	super(context);
04	}
05	@Override
06	protected void onDraw(Canvas canvas) {
07	super.onDraw(canvas);
08	}
09	}

（2）在 onCreate()方法中编写代码实现程序功能

在类 MainActivity 的 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将自定义的 MyView 视图添加到帧布局管理器中，主要代码如表 8-6 所示。

表 8-6　类 MainActivity 中 onCreate()方法的代码

序号	代　　码
01	protected void onCreate(Bundle savedInstanceState) {
02	super.onCreate(savedInstanceState);
03	setContentView(R.layout.activity_main);
04	//获取布局文件中添加的帧布局管理器
05	FrameLayout fl=(FrameLayout)findViewById(R.id.frameLayout1);
06	//将自定义的 MyView 视图添加到帧布局管理器中
07	fl.addView(new MyView(this));
08	}

（3）编写代码定义画笔与绘制几何图形

在 DrawView 的 onDraw()方法中，分别指定画布的背景色、创建一个采用默认设置的画笔、设置该画笔使用抗锯齿功能、设置笔触的宽度、设置填充样式为描边、设置画笔颜色，然后绘制各种几何图形，代码如表 8-7 所示。

表 8-7　定义画笔与绘制几何图形的代码

序号	代　　码
01	protected void onDraw(Canvas canvas) {
02	canvas.drawColor(Color.WHITE);　//指定画布的背景色为白色
03	Paint paint=new Paint();　　//创建采用默认设置的画笔
04	paint.setAntiAlias(true);　　//使用抗锯齿功能
05	paint.setStrokeWidth(3);　　//设置笔触的宽度
06	paint.setStyle(Style.STROKE);　//设置填充样式为描边
07	paint.setColor(Color.BLUE);　　//设置颜色为蓝色
08	//绘制圆形
09	canvas.drawCircle(100, 50, 30, paint);
10	//绘制一条直线

续表

序号	代　　码
11	paint.setColor(Color.RED);
12	paint.setStyle(Style.FILL);
13	canvas.drawLine(150, 50,190, 50, paint);
14	//绘制多条线段组成的折线
15	paint.setColor(Color.BLACK);
16	canvas.drawLines(new float[]{210,30,280,30, 280,30, 245,70, 245,70, 210,30}, paint);
17	//绘制椭圆
18	paint.setStrokeWidth(2); //设置笔触的宽度
19	RectF rectf1=new RectF(60, 100, 150, 160);
20	canvas.drawOval(rectf1,paint);
21	//绘制矩形
22	canvas.drawRect(180, 90, 260, 170, paint);
23	super.onDraw(canvas);
24	}

3．程序运行与功能测试

Android 项目 App0803 的运行结果如图 8-3 所示，完成了多种几何图形的绘制。

【任务 8-4】绘制多种形式的路径

【任务描述】

Android 提供了绘制路径的方法，绘制一条路径可以分为创建路径和绘制定义好的路径两部分，编写代码在屏幕上绘制圆形、三角形、六边形路径以及绕路径的环形文字，其运行结果如图 8-4 所示。



图 8-4　绘制多种形式的路径的运行结果

【知识索引】

（1）Paint 类及 setAntiAlias()、setStrokeWidth()、setStyle()、setColor()、setTextSize()等方法。

（2）Canvas 类及 drawCircle()、setColor()、setStyle()、drawLine()、drawOval()、drawRect()、drawPath()等方法。

（3）Path 类及 addCircle、moveTo()、lineTo()、close()方法。

【实施过程】

1．创建 Android 项目 App0804

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0804。

2．完善布局文件 activity_main.xml 与界面设计

修改项目 App0804 的 res/layout 文件夹下的布局文件 activity_main.xml，将默认添加的 TextView 控件删除，然后将页面的相对布局修改为帧布局类型，用于显示自定义的绘图类。

3．完善类 MainActivity 与实现程序功能

（1）创建一个继承自 android.view.View 类的内部类

在类 MainActivity 中创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法 MyView()和重写 onDraw(Canvas canvas)方法。主体代码如表 8-8 所示。

（2）在 onCreate()方法中编写代码实现程序功能

在类 MainActivity 的 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将自定义的 MyView 视图添加到帧布局管理器中，主要代码如表 8-8 所示。

（3）编写代码定义画笔与绘制几何图形

在 DrawView 的 onDraw()方法中，分别创建一个采用默认设置的画笔、设置该画笔使用抗锯齿功能、设置笔触的宽度、设置填充样式为描边、设置画笔颜色和设置文字大小，然后绘制各种路径，代码如表 8-8 所示。

表 8-8　定义画笔与绘制各种路径的代码

序号	代　　码
01	protected void onDraw(Canvas canvas) {
02	Paint paint = new Paint();　　//创建一个画笔
03	paint.setAntiAlias(true);　　//设置使用抗锯齿功能
04	paint.setStrokeWidth(3);　　//设置笔触的宽度
05	paint.setStyle(Style.STROKE);　　//设置填充方式为描边
06	paint.setColor(Color.BLACK);　　//设置画笔颜色
07	paint.setTextSize(18);　　//设置文字大小
08	//绘制圆形路径
09	Path pathCircle = new Path();　　//创建并实例化一个 path 对象
10	pathCircle.addCircle(70, 70, 40, Path.Direction.CCW);　　//添加逆时针的圆形路径
11	canvas.drawPath(pathCircle, paint);　　//绘制路径
12	//绘制三角形路径
13	Path pathTr = new Path();　　//创建并实例化一个 path 对象
14	pathTr.moveTo(130, 100);　　//设置起始点
15	pathTr.lineTo(180, 50);　　//设置第一条边的结束点，也是第二条边的起始点
16	pathTr.lineTo(230, 100);　　//设置第二条边的结束点，也是第三条边的起始点
17	pathTr.close();　　//闭合路径
18	canvas.drawPath(pathTr, paint);　　//绘制路径
19	//控制空心的六边形
20	Path path1 = new Path();　　//创建并实例化一个 path 对象
21	path1.moveTo(300, 40);　　//设置起始点
22	path1.lineTo(350, 40);　　//设置第一条边的结束点，也是第二条边的起始点

续表

序号	代　　码
23	path1.lineTo(400, 90);　　//设置第二条边的结束点，也是第三条边的起始点
24	path1.lineTo(350, 140);
25	path1.lineTo(300, 140);
26	path1.lineTo(250, 90);
27	path1.close();　　//闭合路径
28	canvas.drawPath(path1, paint);
29	//绘制绕路径的环形文字
30	String str = "业精于勤，荒于嬉，行成于思，毁于随　";
31	Path pathText = new Path();　　//创建并实例化一个 path 对象
32	pathText.addCircle(500, 100, 48, Path.Direction.CW);　　//添加顺时针的圆形路径
33	paint.setStyle(Style.FILL);//设置画笔的填充方式
34	canvas.drawTextOnPath(str, pathText, 0, -18, paint);　　//绘制绕路径文字
35	super.onDraw(canvas);
36	}

4．程序运行与功能测试

Android 项目 App0804 的运行结果如图 8-4 所示，完成了多种路径和环形文字的绘制。

【任务 8-5】绘制 Android 机器人图形

【任务描述】

应用 Android 中绘制几何图形的方法，实现在屏幕上绘制 Android 机器人图形，其运行结果如图 8-5 所示。



图 8-5　绘制 Android 机器人图形的运行结果

【知识索引】

（1）Paint 类及 setAntiAlias()、setColor()、setStrokeWidth()等方法。

（2）RectF 类及 offset()方法

（3）Canvas 类及 drawArc()、drawCircle()、.drawLine()、drawRect()、drawRoundRect()等方法。

【实施过程】

1．创建 Android 项目 App0805

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0805。

2．完善布局文件 activity_main.xml 与界面设计

修改项目 App0805 的 res/layout 文件夹下的布局文件 activity_main.xml，将默认添加的 TextView 控件删除，然后将页面的相对布局修改为帧布局类型，用于显示自定义的绘图类。

3．完善类 MainActivity 与实现程序功能

（1）创建一个继承自 android.view.View 类的内部类

在类 MainActivity 中创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法 MyView()和重写 onDraw(Canvas canvas)方法。主体代码如表 8-9 所示。

（2）在 onCreate()方法中编写代码实现程序功能

在类 MainActivity 的 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将自定义的 MyView 视图添加到帧布局管理器中，主要代码如表 8-9 所示。

（3）编写代码定义画笔与绘制几何图形

在 DrawView 的 onDraw()方法中，首先创建一个采用默认设置的画笔，并设置画笔的相关属性，然后绘制 Android 机器人图形的头、眼睛、天线、身体、胳膊和腿，代码如表 8-9 所示。

表 8-9　定义画笔与绘制 Android 机器人图形的代码

序号	代　　码
01	protected void onDraw(Canvas canvas) {
02	Paint paint=new Paint(); //采用默认设置创建一个画笔
03	paint.setAntiAlias(true); //使用抗锯齿功能
04	paint.setColor(0xFFA4C739); //设置画笔的颜色为绿色
05	//绘制机器人的头
06	RectF rectf_head=new RectF(10, 10, 100, 100);
07	rectf_head.offset(100, 20);
08	canvas.drawArc(rectf_head, -10, -160, false, paint); //绘制弧
09	//绘制眼睛
10	paint.setColor(Color.WHITE); //设置画笔的颜色为白色
11	canvas.drawCircle(135, 53, 4, paint); //绘制圆
12	canvas.drawCircle(175, 53, 4, paint); //绘制圆
13	paint.setColor(0xFFA4C739); //设置画笔的颜色为绿色
14	//绘制天线
15	paint.setStrokeWidth(2); //设置笔触的宽度
16	canvas.drawLine(120, 15, 135, 35, paint); //绘制线
17	canvas.drawLine(190, 15, 175, 35, paint); //绘制线
18	//绘制身体
19	canvas.drawRect(110, 75, 200, 150, paint); //绘制矩形
20	RectF rectf_body=new RectF(110,140,200,160);
21	canvas.drawRoundRect(rectf_body, 10, 10, paint); //绘制圆角矩形
22	//绘制胳膊
23	RectF rectf_arm=new RectF(85,75,105,140);
24	canvas.drawRoundRect(rectf_arm, 10, 10, paint); //绘制左侧的胳膊
25	rectf_arm.offset(120, 0); //设置在 X 轴上偏移 120 像素

续表

序号	代　码
26	canvas.drawRoundRect(rectf_arm, 10, 10, paint);　//绘制右侧的胳膊
27	//绘制腿
28	RectF rectf_leg=new RectF(125,150,145,200);
29	canvas.drawRoundRect(rectf_leg, 10, 10, paint);　//绘制左侧的腿
30	rectf_leg.offset(40, 0);　//设置在 X 轴上偏移 40 像素
31	canvas.drawRoundRect(rectf_leg, 10, 10, paint);　//绘制右侧的腿
32	super.onDraw(canvas);
33	}

4．程序运行与功能测试

Android 项目 App0804 的运行结果如图 8-5 所示，至此完成了 Android 机器人图形的绘制。



Android 平台中的图形显示类是 View，还提供了底层图形类 android.graphics。在 Android 中绘图时，常用到的几个类是 Paint、Canvas、Color、Bitmap 和 BitmaptFactory。Paint 类代表画笔，用来描述图形的颜色和风格，如线宽、颜色、透明度、填充效果等值。使用 Paint 类时，首先需要创建 Paint 类的对象，然后通过该类的成员方法对画笔的默认设置进行更改，例如改变画笔的颜色、线宽等。Canvas 代表画布，通过该类提供的方法，可以绘制各种图形（如矩形、图形、线条等等），通常情况下，要在 Android 绘图，首先要创建一个继承自 View 类的自定义 View，并且在该类中重写 onDraw(Canvas canvas)方法，然后在显示绘图的 Acticity 中添加该自定义 View。Bitmap 称作位图，一般位图的文件格式后缀为 bmp。使用 Bitmap 类提供的方法可以获取位图文件的信息，也可以对位图进行剪切、旋转、缩放等操作，还可以以指定格式保存图像文件。BitmapFactory 类是一个工具类，用于从不同的数据源来解析、创建 Bitmap 对象。Drawable 作为 Android 平台下通用的图形对象，它可以装载常用格式的图像，例如 BMP、GIF、PNG、JPG。

本单元主要介绍了 Bitmap、BitmapFactory、Paint、Canvas、Color、Path 等类的功能、主要方法及其典型应用。通过多个实例学会图片的显示以及图形的绘制方法。



一、填空题

1. Android 中绘图时，常用到的几个类是（　　）、（　　）、（　　）、（　　）和 Color，这些类都存放在（　　）包中。

2．如果 java 程序中需要获取实际的 Drawable 对象，则可以调用 Resources 的（　　）方法获取。

3．如果需要获取 BitmapDrawable 所包装的 Bitmap 对象，则可以调用 BitmapDrawable 的

（　　）方法。

4．Android 的绘图应该继承（　　）组件，并重写它的 onDraw（Canvas canvas）方法即可。

5．类（　）主要用于设置绘制风格，包括画笔颜色、画笔笔触粗细、填充风格等。

6．对图片添加旋转、缩放等特效需要使用（　　）类。

二、选择题

1．Canvas 类可以用来实现各种图形的绘制工作，其中可以绘制矩形的方法是（　　）。

A．drawArc()　　B．drawCircle()　　C．drawOval()　　D．drawRect()

2．进行图形绘制时需要调用 Canvas 类的方法，以下哪个方法可以用来绘制三角形的三条边？（　　）。

A．drawPoint　　B．drawLine　　C．drawCircle　　D．drawRect

3．Paint 类用来描述画笔，Paint 不能设定以下哪个属性？（　　）。

A．文字大小　　B．坐标位置　　C．抗锯齿效果　　D．文字对齐方式

4．以下哪些方法可以获取资源中的位图？（　　）。（多选题）

A．BitmapDrawable　　B．BitmapFactory　　C．Canvas　　D．Paint

5．Bitmap 提供了一些静态方法来创建新的 Bitmap 对象，下列哪些属于其常用静态方法的是（　　）。（多选题）

A．createBitmap(Bitmap source，int x，int y，int width，int height)

B．createScaleBitmap（Bitmap src，int dstWidth，int dstHeight，boolean filter）

C．createBitmap（int width，int height，Bitmap.Config config）

D．createBitmap（Bitmap source，int x，int y，int width，int height，Matrix m，boolean filter）

三、简答题

1．简述 Bitmap 类常用的方法及其主要功能。

2．简述位图缩放的主要实现方法。

3．简述 Canvas 类用来实现各种图形绘制工作的方法。

4．简述 Pain 类的常用属性设置方法。

单元 9

Android 的音频与视频应用程序设计

Android 的 SoundPool 类是 android.media 包里提供的一个用来播放声音文件的类，可以支持同时播放多个声音文件，可以控制每个文件的循环次数。用 soundpool 可以播一些短的反应速度要求高的声音，例如游戏中的声音。在 Android 中通常用 MediaPlayer 来播放一些媒体文件，对于音频文件来说只需直接使用 MeidaPlayer 结合几句代码即可。但是对于视频文件来说稍微复杂一些，单独的 MediaPlayer 只能播放音频文件，要想播放视频还需要 SurfaceView 来配合显示画面。Android 播放视频主要使用 VideoView 或者 SurfaceView，其中 VideoView 组件播放视频最简单，它将视频的显示和控制集于一身，因此，借助它就可以完成一个简易的视频播放器。而对于 SurfacView 而言，在 Android 中采用了显示与控制分离机制，即 SurfaceView 只是负责显示画面，而不负责控制视频流，因此还需要 SurfaceHolder 来控制视频流。本单元主要探讨 Android 的音频与视频相关类的功能及实现方法。

教学导航

教学目标	（1）熟悉 Android 中播放声音的主要类 SoundPool 和 MediaPlayer （2）熟悉 Android 中播放视频的主要类 VideoView 和 SurfaceView （3）学会使用 SoundPool 类播放音频 （4）学会使用 MediaPlayer 类播放本地音频和网络音频 （5）学会使用 VideoView 控件播放本地视频 （6）学会使用 MediaPlayer 类和 SurfaceView 控件播放本地视频
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	8 课时

知识导读

9.1 SoundPool 类与播放音频

开发 Android 软件中我们可能经常需要播放多媒体声音文件，一般使用 MediaPlayer 类但该类占用资源较多，对于游戏等应用可能不是很适合。SoundPool（声音池）主要播放一些较短的声音片段，可以从程序的资源或文件系统加载，相比 MediaPlayer 类可以使用较少的 CPU 资源和具有较短的反应延迟。SoundPool 类和其他声音播放类相比，其特点是可以自行设置声

音的品质、音量、播放比率等参数等。并且它可以同时管理多个音频流，每个流都有独自的 ID，对某个音频流的管理都是通过 ID 进行的。由于 SoundPool 最大只能申请 1MB 的内存空间，这就意味着我们只能播放很短的声音片段，而不是用它来播放歌曲或者做游戏背景音乐。

1．创建一个 SoundPool

创建一个 SoundPool 对象的构造方法为：

```
public SoundPool(int maxStream, int streamType, int srcQuality)
```

构造方法中包含 3 个参数，其中 maxStream 表示同时播放的流的最大数量，streamType 表示流的类型，一般为 STREAM_MUSIC；srcQuality 表示采样率转化质量，使用 0 作为默认值。

示例代码如下所示：

```
SoundPool soundPool = new SoundPool(3, AudioManager.STREAM_MUSIC, 0);
```

表示创建了一个最多支持 3 个流同时播放的，类型标记为音乐的 SoundPool。

2．载入音频流

（1）从系统资源载入音频流

```
int load(Context context, int resId, int)
```

（2）从 FileDescriptor 对象载入音频流

```
int load(FileDescriptor fd, long offset, long length, int priority)
```

其中参数 priority 表示优先级。

（3）从完整文件路径名载入音频流

```
int load(String path, int priority)
```

一般把多个声音放到 HashMap 中去，再载入，示例代码如下：

```
soundPool = new SoundPool(4, AudioManager.STREAM_MUSIC, 100);
soundPoolMap = new HashMap<Integer, Integer>( );
soundPoolMap.put(1, soundPool.load(this, R.raw.dingdong, 1));
```

3．播放控制

有以下几个方法可用于控制播放控制：

① final int play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)：用于播放指定音频的音效，并返回一个 streamID。

参数 leftVolume 和 rightVolume 表示左右音量，priority 表示流的优先级，值越大优先级高，影响当同时播放数量超出了最大支持数时 SoundPool 对该流的处理；loop 表示循环播放的次数，0 值为播放一次，-1 为无限循环，其他值为播放 loop+1 次（例如，3 为一共播放 4 次），rat 表示播放的速率，范围为 0.5～2.0(0.5 为一半速率，1.0 为正常速率，2.0 为两倍速率)。

示例代码：sp.play(soundId, 1, 1, 0, 0, 1);

② final void pause(int streamID)：暂停指定播放流的音效（streamID 应通过 play()返回），这里的 streamID 和 soundID 均在构造 SoundPool 类的第一个参数中指明了总数量，而 id 从 0

开始。

③ final void resume(int streamID)：继续播放指定播放流的音效（streamID 应通过 play()返回）。

④ final void stop(int streamID)：终止指定播放流的音效（streamID 应通过 play()返回）。

④ final void setLoop(int streamID, int loop)：设置指定播放流的循环。

⑤ final void setVolume(int streamID, float leftVolume, float rightVolume)：设置指定播放流的音量。

⑥ final void setPriority(int streamID, int priority)：设置指定播放流的优先级。

⑦ final void setRate(int streamID, float rate)：设置指定播放流的速率。

⑧ final boolean unload(int soundID)：卸载一个指定的音频资源。

⑨ final void release()：释放 SoundPool 中的所有音频资源。

【注意】:

① play()函数传递的是一个 load()返回的 soundID，指向一个被记载的音频资源。如果播放成功则返回一个非 0 的 streamID，指向一个成功播放的流；同一个 soundID 可以通过多次调用 play()而获得多个不同的 streamID (只要不超出同时播放的最大数量)。

② pause()、resume()和 stop()是针对播放流操作的，传递的是 play()返回的 streamID。play()中的 priority 参数只在同时播放流的数量超过了预先设定的最大数量时起作用，管理器将自动终止优先级低的播放流。如果存在多个同样优先级的流，再进一步根据其创建事件来处理，最新创建的流将被终止。

③ 一个 SoundPool 可以通过 load()函数管理多个音频资源，成功则返回非 0 的 soundID。通过 play()函数同时播放多个音频，成功则返回非 0 的 streamID。

④ 当设置为无限循环时，需要手动调用 stop()来终止播放。

⑤ 无论如何，程序退出时，手动终止播放并释放资源是必要的。

9.2 MediaPlayer 类与播放音频

Android 提供了 MediaPlayer 类来完成对音频和视频的播放，通过 MediaPlayer 类我们可以播放应用程序资源文件、本地文件或者通过 URL 获得的流媒体。

1．MediaPlayer 类的常用方法

MediaPlayer 支持多种格式的音频文件并提供非常全面的控件方法，从而使得播放音乐变得十分简单。MediaPlayer 类控制音频的常用方法如表 9-1 所示。

表 9-1　MediaPlayer 类的常用方法

方 法 声 明	功 能 描 述
create(Context context, Uri uri)	静态方法，通过 Uri 创建一个多媒体播放器
create(Context context, int resid)	静态方法，通过资源 ID 创建一个多媒体播放器
create(Context context, Uri uri, SurfaceHolder holder)	静态方法，通过 Uri 和指定 SurfaceHolder（抽象类）创建一个多媒体播放器
getCurrentPosition()	返回 Int，获取当前播放位置
getDuration()	返回 Int，获取载入音频文件的时长

续表

方 法 声 明	功 能 描 述
getVideoHeight()	返回 Int，获取视频的高度
getVideoWidth()	返回 Int，获取视频的宽度
isLooping()	返回 boolean，是否循环播放
isPlaying()	返回 boolean，判断是否正在播放
pause()	无返回值，暂停播放
prepare()	无返回值，准备同步，用于播放本地音乐
prepareAsync()	无返回值，准备异步，用于播放网络音乐
release()	无返回值，释放 MediaPlayer 对象相关的资源
reset()	无返回值，重置 MediaPlayer 对象
seekTo(int msec)	无返回值，指定播放的位置（以毫秒为单位的时间）
setAudioStreamType(int streamtype)	无返回值，指定流媒体的类型，取值为 STREAM_MUSIC 表示音乐，STREAM_RING 表示响铃，STREAM_ALARM 表示闹钟，STREAM_NOTIFICTION 表示提示音
setDataSource(String path)	无返回值，根据路径设置多媒体数据来源
setDataSource(FileDescriptor fd, long offset, long length)	无返回值，根据 FileDescriptor 设置多媒体数据来源
setDataSource(FileDescriptor fd)	无返回值，根据 FileDescriptor 设置多媒体数据来源
setDataSource(Context context, Uri uri)	无返回值，根据 Uri 设置多媒体数据来源
setDisplay(SurfaceHolder sh)	无返回值，设置用 SurfaceHolder 来显示多媒体
setLooping(boolean looping)	无返回值，设置是否循环播放
setScreenOnWhilePlaying(boolean screenOn)	无返回值，设置是否使用 SurfaceHolder 显示
setVolume(float leftVolume, float rightVolume)	无返回值，设置音量
start()	无返回值，开始播放
stop()	无返回值，停止播放

对 MediaPlayer 类常用方法的使用说明如下。

（1）获得 MediaPlayer 实例的方式

可以使用直接 new 的方式：MediaPlayer mp = new MediaPlayer();

也可以使用 create 的方式：MediaPlayer mp = MediaPlayer.create(this, R.raw.test);

这时就不用调用 setDataSource 了。

（2）设置音频数据源的方式

初始化时调用 MediaPlayer 的 Create 静态方法，可以有四种方法设置音频的数据源。

（1）以资源描述符表示播放源为自带的 resource 资源

将音频文件添加到资源结构中的 res/raw 文件夹中。

```
MediaPlayer player = MediaPlayer.create(getApplicationContext(), R.raw.music);
```

也可以写成

```
MediaPlayer player = MediaPlayer.create(this , R.raw.music);
```

（2）以 file://开头的本地文件 URI 表示播放源为存储在 SD 卡或其他文件路径下的媒体文件

```
MediaPlayer player = MediaPlayer.create(getApplicationContext(),
                     Uri.parse (“file:///sdcard/music/ test.mp3”));
```

（3）以 URL 为地址的网络文件 URI 表示播放源为网络上的媒体文件

```
MediaPlayer player = MediaPlayer.create(getApplicationContext(),
                     Uri.parse ("http://www.music.net/ test.mp3"));
```

（4）根据内容提供器提供的 URI 设置播放源

```
MediaPlayer player = MediaPlayer.create(getApplicationContext(), Settings.
System.DEFAULT_RINGTONE_URI);
```

也可以使用 MediaPlayer 的 setDataSource 成员方法设置数据源。当调用此方法作为初始化方式时，在调用 setDataSource 之后，应当调用 prepare()成员方法。代码如下：

```
MediaPlayer player = new MediaPlayer();
player.setDataSource("/sdcard/test.mp3");
player.prepare();
```

对于网络上的媒体文件应写成

```
player.setDataSource("http://www.music.net/test.mp3");
```

（3）对播放器的主要控制方法

Android 通过控制播放器的状态来控制媒体文件的播放，其中：prepare()和 prepareAsync()提供了同步和异步两种方式设置播放器进入 prepare 状态。prepare()方法执行完毕后 MediaPlayer 进入 prepared 状态。prepareAsync()方法执行完毕后 MediaPlayer 进入 preparing 状态。prepareAsync()方法一般用于加载网络音频文件等耗时的场景，而加载本地音频的时候一般使用 prepare()就可以了。

【注意】: 在不合适的状态下调用 prepare()和 prepareAsync()方法会抛出 IllegalStateException 异常。当 MediaPlayer 对象处于 Prepared 状态的时候，可以调整音频/视频的属性，如音量，播放时是否一直亮屏，循环播放等。

如果 MediaPlayer 实例是由 create 方法创建的，那么第一次启动播放前不需要再调用 prepare()了，因为 create 方法里已经调用过了。start()是真正启动文件播放的方法，pause()和 stop()比较简单，起到暂停和停止播放的作用，seekTo()是定位方法，可以让播放器从指定的位置开始播放，需要注意的是该方法是个异步方法，也就是说该方法返回时并不意味着定位完成，尤其是播放网络文件，真正定位操作完成时内部的播放引擎会调用客户端提供的 OnSeekComplete.onSeekComplete()回调方法，如果需要可以调用 setOnSeekCompleteListener (OnSeekCompleteListener)方法设置监听器来处理。release()可以释放播放器占用的资源，一旦确定不再使用播放器时应当尽早调用它释放资源。reset()可以使播放器从 Error 状态中恢复过来，重新回到 Idle 状态。

（4）设置播放器的监听器

MediaPlayer 提供了一些设置不同监听器的方法来更好地对播放器的工作状态进行监听，

以期及时处理各种情况，MediaPlayer 类控制音频的常用监听事件如下所示。

① setOnBufferingUpdateListener(MediaPlayer.OnBufferingUpdateListener listener)：监听网络流媒体的缓冲。

② setOnCompletionListener(MediaPlayer.OnCompletionListener listener)：监听网络流媒体播放是否结束。

③ setOnErrorListener(MediaPlayer.OnErrorListener listener)：监听设置错误信息。

④ setOnVideoSizeChangedListener(MediaPlayer.OnVideoSizeChangedListener listener)：监听视频尺寸。

2．MediaPlayer 对象的生命周期与播放控制

使用 new 操作符创建一个新的 MediaPlayer 对象或是对已有对象调用 reset()方法之后 MediaPlayer 对象处于 Idle 状态。调用 setDataSource()方法会使处于 Idle 状态的对象转变为 Initialized 状态。在开始播放之前，MediaPlayer 对象必须要进入 Prepared 状态，有同步和异步两种方式可以使 MediaPlayer 对象进入 Prepared 状态。

准备好 MediaPlayer 后，想要开始播放，必须调用 start()方法。当此方法成功返回时，MediaPlayer 的对象处于 Started 状态。

isPlaying()方法可以被调用来测试某个 MediaPlayer 对象是否在 Started 状态。而对一个已经处于 Started 状态的 MediaPlayer 对象调用 start()方法并没有影响。

调用 pause()方法并返回时，播放可以被暂停会使 MediaPlayer 对象进入 Paused 状态。调用 start()方法会让一个处于 Paused 状态的 MediaPlayer 对象从之前暂停的地方恢复播放。当调用 start()方法返回的时候，MediaPlayer 对象的状态会又变成 Started 状态。

【注意】：Started 与 Paused 状态的相互转换在内部播放引擎中是异步的，所以可能需要一点时间在 isPlaying()方法中更新状态，若在播放流内容，这段时间可能会有几秒钟。

调用 stop()方法会停止播放，并且还会让一个处于 Started，Paused，Prepared 或 PlaybackCompleted 状态的 MediaPlayer 对象进入 Stopped 状态。

调用 seekTo()方法可以调整播放的位置。seekTo(int)方法是异步执行的，所以它可以马上返回，但是实际的定位播放操作可能需要一段时间才能完成，尤其是播放流形式的音频/视频。

【注意】：seekTo(int)方法也可以在其他状态下调用，例如 Prepared，Paused 和 PlaybackCompleted 状态。此外，目前的播放位置，实际可以调用 getCurrentPosition()方法得到，它可以帮助如音乐播放器的应用程序不断更新播放进度。

3．释放播放资源

当播放到流的末尾，播放就完成了。如果调用了 setLooping(boolean)方法开启了循环模式，那么这个 MediaPlayer 对象会重新进入 Started 状态。

在播放结束时，应调用 MediaPlayer 的 release()方法，释放播放程序所占用的资源。

代码如下：

```
private void release() {
    if(player != null) {
      player.stop();
      player.release();
      player = null;
    }
}
```

4．视频播放

视频播放与音频播放在 MediaPlayer 中基本相同，但与音频不同的是，视频播放必须考虑到 MediaPlayer 的视频显示方法。下面简要描述可以用于视频播放显示的两种方法。

（1）使用 VideoView 播放显示视频

由于 VideoView 继承了 SurfaceView，并且实现了 MediaController.MediaPlayerControl。使得 VideoView 控件包含了一个视频显示的界面，该界面可以管理 MediaPlayer 以控制视频的播放，同样，VideoView 也支持对 MediaPlayer 组件所支持的本地文件或者流式文件播放。

（2）使用 SurfaceHolder 播放显示视频

虽然 VideoView 可以很容易地播放视频，但播放位置和播放大小并不受控制，因此，想要使用 MediaPlayer 播放视频，需要使用 SurfaceView 来播放视频。对于显示视频内容则需要调用 MediaPlayer 的 setDisplay()方法分配一个 SurfaceHolder 对象，否则，视频将不会显示。

9.3 VideoView 类与播放视频

播放视频文件与播放音频文件类似，与音频播放相比，视频的播放需要使用视频播放组件将影像展示出来。在 Android 中，播放视频主要使用 VideoView 或者 SurfaceView，其中 VideoView 组件播放视频最简单，它将视频的显示和控制集于一身，因此，借助它就可以完成一个简易的视频播放器。

1．使用 VideoView 类播放视频的主要步骤

使用 VideoView 类播放视频的主要步骤如下所示：

（1）在界面布局文件中定义 VideoView 组件，或在程序中创建 VideoView 组件。

（2）调用 VideoView 的如下两个方法来加载指定的视频。

① setVidePath(String path)：加载 path 文件代表的视频。

② setVideoURI(Uri uri)：加载 uri 所对应的视频。

（3）调用 VideoView 的 start()、stop()、psuse()方法来控制视频的播放。

2．VideoView 类的常用方法

VideoView 类的用法与 MediaPlayer 类类似，也提供了一些控制视频播放方法，如表 9-2 所示。

表 9-2　VideoView 类的常用方法

方 法 名 称	功 能 描 述
setVideoPath()	设置要播放视频文件的位置
setVideoURI()	设置要播放视频的源地址
start()	开始或继续播放视频
pause()	暂停播放视频
resume()	将视频从头开始播放
seekTo()	从指定位置开始播放视频
isPlaying()	判断当前是否正在播放视频
getCurrentPosition	获取当前播放位置

续表

方 法 名 称	功 能 描 述
getDuration()	获取载入的视频文件的时长
getBufferPercentage()	获取缓冲的百分比
resolveAdjustedSize()	调整视频显示大小
setMediaController()	设置播放控制器模式（播放进度条）

当视频文件播放完时触发 setOnCompletionListener 事件。

3．创建 VideoView 对象

不同于音乐播放器，视频播放需要界面中显示影像，因此需要在 XML 布局文件中添加 VideoView 控件，示例代码如下所示：

```
<VideoView
   android:id="@+id/videoView"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_gravity="center" />
```

4．视频的播放

使用 VideoView 播放视频和音频一样，既可以播放本地视频，也可以播放网络中的视频。

（1）播放本地视频

```
VideoView video1=(VideoView)findViewById(R.id.videoView);
video1.setVideoPath("/mnt/sdcard/v01.mp4");
video1.start();
```

（2）播放网络视频

```
VideoView video2=(VideoView)findViewById(R.id.videoView);
Uri uri = Uri.parse(Environment.getExternalStorageDirectory().getPath()+"/
v02.mp4");
video2.setVideoURI(uri);
video2.start();
```

5．为 VideoView 添加控制器

VideoView 通过与 MediaController 类结合使用，开发者可以不用自己控制播放与暂停。使用 VideoView 播放视频时可以为它添加一个控制器 MediaController，它是一个包含媒体播放器 MediaPlayer()控件的视图，主要实现对 VideoView 的播放予以控制。MediaController 包含了一些典型按钮，如播放/暂停（Play/Pause）、回退（Rewind）、快进（Fast Forward）与进度滑动器（Progress Slider）。

播放本地视频完整的示例代码如下所示：

```
VideoView video1;
video1=(VideoView)findViewById(R.id. videoView);
MediaController mediaco=new MediaController(this);
File file=new File("/mnt/sdcard/v01.mp4");
if(file.exists()){
   video1.setVideoPath(file.getAbsolutePath());
```

```
        video1.setMediaController(mediaco);
        //VideoView 与 MediaController 进行关联
        mediaco.setMediaPlayer(video1);
        video1.requestFocus();
        //让 VideiView 获取焦点
    }
```

播放网络视频完整的示例代码如下所示：

```
    Uri uri = Uri.parse(Environment.getExternalStorageDirectory().getPath()+"/
v02.mp4");
    VideoView video2 = (VideoView)this.findViewById(R.id.videoView);
    video2.setMediaController(new MediaController(this));
    video2.setVideoURI(uri);
    video2.start();
    video2.requestFocus();
```

9.4 MediaPlayer 与 SurfaceView 联合播放视频

使用 VideoView 播放视频虽然方便，但不易于扩展，当开发者需要根据需求自定义视频播放器时，使用 VideoView 就会很麻烦。为此，Android 系统还提供了另一种播放视频的方法，就是 MediaPlayer 与 SurfaceView 一起联合使用。MediaPlayer 可以播放视频，只不过它在播放视频时没有图像输出，因此需要使用 SurfaceView 组件。

SurfaceView 继承自 View 类，是用于显示图像的组件。SurfaceView 的最大特点就是它的双缓冲技术，所谓双缓冲技术就是它内部有两个线程，例如线程 A 和线程 B，当线程 A 更新界面时，线程 B 进行后台计算操作，当两个线程都完成了各自的任务时，它们会互相交换，两个线程就这样无限循环交替更新和计算。由于 SurfaceView 的这种特性可以避免任务繁重而造成主线程阻塞，从而提高了程序的反应速度，因此在游戏开发中多使用 SurfaceView，例如游戏中的背景、人物和动画等场合。

1．在界面中添加 SurfaceView 控件

SurfaceView 是一个控件，在布局文件中添加该控件的代码如下所示：

```
    <SurfaceView
        android:id="@+id/surface"
        android:layout_height="220dip"
        android:layout_width="320dip">
        android:layout_gravity="center"
        android:layout_weight="0.25"
    </SurfaceView>
```

2．获取界面显示容器并设置类型

布局文件创建完成，在代码中通常 id 找到该控件并获取到 SurfaceView 控件的容器 SurfaceHolder，示例代码如下所示：

```
    MediaPlayer player;
    SurfaceView surface;
    SurfaceHolder surfaceHolder;
    private String filename;
    private int prosition=0;
```

```
    private final static String TAG="VodeoPlayActivity";
    surface=(SurfaceView)findViewById(R.id.surface);
    surfaceHolder=surface.getHolder();//SurfaceHolder 是 SurfaceView 的控制接口
    //该类实现了 SurfaceHolder.Callback 接口，回调参数直接 this
    surfaceHolder.addCallback(this);
    surfaceHolder.setFixedSize(320, 220);   //显示的分辨率,不设置为视频默认
    surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    //Surface 类型
```

SurfaceHolder 是一个接口类型，它用于维护和管理显示的内容，也就相当于 SurfaceView 的管理器。通过 SurfaceHolder 对象控制 SurfaceView 的大小和像素格式，监视控件中内容的变化。

【注意】：在进行游戏开发时，使用 SurfaceView 需要开发者手动创建维护两个线程进行双缓冲区的管理，而播放视频时使用 MediaPlayer 框架，它是通过底层代码去管理和维护音视频文件。因此，需要添加 SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS 参数不让 SurfaceView 自己维护双缓冲区，而是交给 MediaPlayer 底层去管理。

3．为 SurfaceHolder 添加回调

如果在 onCreate()方法执行时，SurfaceHolder 还没有完全创建好。此时播放视频会出现异常。因此，需要添加 SurfaceHolder 的回调函数 Callback，在 SurfaceCreated()方法中执行视频的插入。示例代码如下所示。

```
    private void play() {
       try {
           File file=new File(Environment.getExternalStorageDirectory(),filename);
           mediaPlayer.reset();     //重置为初始状态

          mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
          mediaPlayer.setDisplay(surfaceView.getHolder());
          mediaPlayer.setDataSource(file.getAbsolutePath());  //设置路径
          mediaPlayer.prepare();  //缓冲
          mediaPlayer.start();    //播放
        } catch (Exception e) {
          Log.e(TAG, e.toString());
          e.printStackTrace();
         }
    }
    private final class SurceCallBack implements SurfaceHolder.Callback{
        //画面创建
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
            if(prosition>0&&filename!=null){
               play();
               mediaPlayer.seekTo(prosition);
               prosition=0;
            }
        }
        //画面修改

        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,
int height){
```

```
            // TODO Auto-generated method stub
        }
        //画面销毁
        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
            if(mediaPlayer.isPlaying()){
              prosition=mediaPlayer.getCurrentPosition();
              mediaPlayer.stop();
            }
        }
    }
```

Callback 接口有 3 个回调方法：surfaceCreated()、surfaceChanged()、surfaceDestroyed()。

4．创建 MediaPlayer

使用 MediaPlayer 播放音频与播放视频的步骤类似，唯一不同的是，播放视频需要将视频显示在 SurfaceView 界面上，因此就需要将 SurfaceView 与 MediaPlayer 进行关联，示例代码如下所示：

```
//必须在 surface 创建后才能初始化 MediaPlayer，否则不会显示图像
MediaPlayer player=new MediaPlayer();
player.setAudioStreamType(AudioManager.STREAM_MUSIC);
player.setDisplay(surfaceHolder);
//设置显示视频显示在 SurfaceView 上
try {
   player.setDataSource("/sdcard/v03.mp4");
   player.prepare();
  } catch (Exception e) {
    e.printStackTrace();
}
```

上述代码中，player.setDisplay(surfaceHolder)表示将播放的视频显示在 SurfaceView 的容器中播放，如果视频很大，可以使用 prepareAsync()方法异步准备。



【任务 9-1】使用 SoundPool 类播放音频

【任务描述】



图 9-1　使用 SoundPool 类播放音频的程序运行结果

Android 提供了播放音频的 SoundPool 类，该类适合在应用程序中播放按键音和消息提示音等，也适合在游戏中实现密集而短暂的声音。编写程序实现使用 SoundPool 类播放音频。程序的运行结果如图 9-1 所示。

【知识索引】

（1）SoundPool 类、put()方法、get()方法、play()方法。

（2）HashMap 类、AudioManager 类、STREAM_MUSIC 常量。

（3）OnClickListener 类、setOnClickListener()方法、onClick()方法。

【实施过程】

1．创建 Android 项目 App0901 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0901。

在项目 App0901 的文件夹 res 中创建子文件夹“raw”，并将本任务所需的音频文件导入或复制到 res/ raw 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

修改项目 App0901 的 res/layout 文件夹下的布局文件 activity_main.xml，将布局方式设置为水平线性布局，并将默认添加的 TextView 控件删除，然后添加两个 Button 控件，将 Button 控件的 id 属性分别设置为 button1 和 button2、text 属性设置为“门铃声”和“风铃声”。

3．完善类 MainActivity 与实现程序功能

（1）声明与创建对象

在主活动 MainActivity 中，分别声明 SoundPool 对象和创建 HashMap 对象，具体代码如下所示：

```
private SoundPool soundpool;    //声明一个 SoundPool 对象
//创建一个 HashMap 对象
private HashMap<Integer, Integer> soundmap = new HashMap<Integer, Integer>();
```

（2）在 onCreate()方法中编写代码实现程序功能

在类 MainActivity 的 onCreate()方法中，首先获取布局文件中添加的“门铃声”按钮和“风铃声”按钮，然后实例化 SoundPool 对象，再将要播放的全部音频流添加到 HashMap 对象中，主要代码如表 9-3 所示。

表 9-3　类 MainActivity 的 onCreate()方法的部分代码

序号	代　　码
01	Button notify = (Button) findViewById(R.id.button1);　　//获取“门铃声”按钮
02	Button ringout = (Button) findViewById(R.id.button2);　　//获取“风铃声”按钮
03	//创建一个 SoundPool 对象，该对象可以容纳 2 个音频流
04	soundpool = new SoundPool(2,AudioManager.STREAM_MUSIC, 0);
05	//将要播放的音频流添加到 HashMap 对象中
06	soundmap.put(1, soundpool.load(this, R.raw.notify, 1));
07	soundmap.put(2, soundpool.load(this, R.raw.ring, 1));

（3）编写代码为按钮添加单击事件监听器

编写代码分别为“门铃声”按钮和“风铃声”按钮添加单击事件监听器，在重写的 onClick()方法中播放指定音频，实现代码如表 9-4 所示。

表 9-4　onCreate()方法中单击事件过程的代码

序号	代　　码
01	//为各按钮添加单击事件监听器
02	notify.setOnClickListener(new OnClickListener() {
03	@Override
04	public void onClick(View v) {
05	soundpool.play(soundmap.get(1), 1, 1, 0, 0, 1);　　//播放指定的音频
06	}
07	});
08	ringout.setOnClickListener(new OnClickListener() {
09	@Override
10	public void onClick(View v) {
11	soundpool.play(soundmap.get(2), 1, 1, 0, 0, 1);　　//播放指定的音频
12	}
13	});

4．程序运行与功能测试

Android 项目 App0901 的运行结果如图 9-2 所示，单击【门铃声】按钮或【风铃声】按钮将播放相应音乐。



图 9-2　App0901 的运行结果

【任务 9-2】使用 MediaPlayer 类播放本地音频

【任务描述】

编写程序制作一款简易音乐播放器，要求实现播放、暂停/继续和停止功能。

【知识索引】

（1）MediaPlayer 类及 start()、isPlaying()、pause()、stop()、prepare()、release()等方法。

（2）Intent 类、Activity 类及 stopService()方法。

（3）try…catch 语句。

【实施过程】

1．创建 Android 项目 App0902 与资源准备

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0902。

在项目 App0902 的文件夹 res 中创建子文件夹“raw”，并将本任务所需的音乐文件导入或复制到 res/ raw 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

修改项目 App0902 的 res/layout 文件夹下的布局文件 activity_main.xml，将布局方式设置为水平线性布局，并将默认添加的 TextView 控件删除，然后添加 3 个 Button 控件，并设置各个控件的属性。修改完善后布局文件 activity_main.xml 的代码如表 9-5 所示。

表 9-5　项目 App0901 中布局文件 activity_main.xml 的代码

序号	布 局 代 码
01	<LinearLayout>
02	<EditText
03	android:id="@+id/et_mediapath"
04	android:layout_width="match_parent"
05	android:layout_height="wrap_content"
06	android:text="m01.mp3"/>
07	<LinearLayout
08	android:layout_width="match_parent"
09	android:layout_height="wrap_content"
10	android:orientation="horizontal"
11	android:layout_gravity="center_horizontal"
12	android:gravity="center">
13	<Button
14	android:id="@+id/btnPlay"
15	android:layout_width="0dp"
16	android:layout_weight="1"
17	android:layout_height="wrap_content"
18	android:text="播放"
19	android:gravity="center"
20	android:drawableTop="@android:drawable/ic_media_play"
21	android:drawablePadding="3dp"/>
22	<Button
23	android:id="@+id/btnPause"
24	android:layout_width="0dp"
25	android:layout_weight="1"
26	android:layout_height="wrap_content"
27	android:drawableTop="@android:drawable/ic_media_pause"
28	android:drawablePadding="3dp"
29	android:gravity="center"
30	android:text="暂停" />
31	<Button
32	android:id="@+id/btnStop"
33	android:layout_width="0dp"
34	android:layout_weight="1"
35	android:layout_height="wrap_content"
36	android:drawablePadding="3dp"
37	android:drawableTop="@android:drawable/ic_media_pause"
38	android:gravity="center"
39	android:text="停止" />
40	</LinearLayout>
41	</LinearLayout

表 9-5 中的 Button 控件使用到属性 android:drawableTop，该属性用于设置文字上方的小图片。

3．完善 MainActivity 类与实现程序功能

（1）声明与创建对象

在主活动 MainActivity 中，声明多个对象。具体代码如下所示：

```
private boolean isPause = false;   //是否暂停
private Button play;
private Button pause;
private Button stop;
private Button close;
private Button exit;
private Intent intent;
private MediaPlayer player = new MediaPlayer();   //定义音乐播放对象
```

（2）在 onCreate()方法中编写代码实现程序功能

类 MainActivity 的 onCreate()方法的主要代码如表 9-6 所示。第 05 行创建装载音频资源（res/raw/m01.mp3）的 MediaPlayer 对象，第 08～12 行获取布局管理器中的按钮控件，第 14～19 行为 MediaPlayer 对象添加事件监听器，用于当音乐播放完毕后，能重新开始播放音乐，第 21～23 行为按钮添加单击事件监听器。

表 9-6　类 MainActivity 的 onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	player=MediaPlayer.create(this,R.raw.m01);
06	//player.setDataSource("/sdcard/m01.mp3");
07	//获取布局中的所有按钮控件
08	play = (Button) findViewById(R.id.btnPlay);
09	pause = (Button) findViewById(R.id.btnPause);
10	stop = (Button) findViewById(R.id.btnStop);
11	close = (Button) findViewById(R.id.btnClose);
12	exit = (Button) findViewById(R.id.btnExit);
13	// 为 MediaPlayer 对象添加完成事件监听器
14	player.setOnCompletionListener(new OnCompletionListener() {
15	@Override
16	public void onCompletion(MediaPlayer mp) {
17	player.start();　　//重新开始播放
18	}
19	});
20	//设置按钮的单击事件监听器
21	play.setOnClickListener(mylistener);
22	pause.setOnClickListener(mylistener);
23	stop.setOnClickListener(mylistener);
24	}

（3）编写代码自定义单击事件监听器对象

编写代码定义单击事件监听器对象，其实现代码如表 9-7 所示。

表 9-7　自定义单击事件监听器对象的代码

序号	代　　码
01	OnClickListener mylistener;
02	{
03	mylistener = new OnClickListener() {
04	@Override
05	public void onClick(View v) {
06	intent = new Intent("android.intent.action.MAIN");
07	switch (v.getId()) {
08	case R.id.btnPlay:
09	player.start();　　//开始播放
10	if (isPause) {
11	pause.setText("暂停");
12	isPause = false;　　//设置暂停标记变量的值为 false
13	}
14	break;
15	case R.id.btnPause:
16	//音乐暂停
17	if (player.isPlaying() && !isPause) {
18	player.pause();　　//暂停播放;
19	isPause = true;
20	((Button) v).setText("继续");
21	} else {
22	player.start();　　//继续播放
23	((Button) v).setText("暂停");
24	isPause = false;
25	}
26	break;
27	case R.id.btnStop:
28	if (player != null) {
29	player.stop();
30	try {
31	// 在调用 stop 后如果需要再次通过 start 进行播放,需要之前调用 prepare 函数
32	player.prepare();
33	} catch (IOException ex) {
34	ex.printStackTrace();
35	}
36	}
37	break;
38	case R.id.btnClose:　　// close activity

续表

```
39                              MainActivity.this.finish();
40                              break;
41                          case R.id.btnExit:          // stopService
42                              stopService(intent);
43                              MainActivity.this.finish();
44                              break;
45                          default:
46                              break;
47                      }
48                  }
49              };
50  }
```

表 9-7 中第 39 行只是执行了 this.finish();语句，关闭了本 Activity 窗体，service 并没有被关掉，音乐依然会继续在后台播放。

第 42 行先调用了 stopService(intent);语句，关闭了 service 服务，在 Service 中会调用 onDestroy()停止并释放音乐资源后才执行 this.finish();语句，关闭了本 Activity 窗体。

（4）重写 Activity 的 onDestroy()方法

onDestroy()方法的代码如表 9-8 所示，用于在当前 Activity 销毁时，停止正在播放的视频，并释放 MediaPlayer 所占用的资源。

表 9-8　Activity 的 onDestroy()方法的代码

序号	代　码
01	@Override
02	protected void onDestroy() {
03	if(player.isPlaying()){
04	player.stop();　//停止音频的播放
05	}
06	player.release();　//释放资源
07	super.onDestroy();
08	}

4．程序运行与功能测试



图 9-3　App0902 的运行结果

Android 项目 App0902 的运行结果如图 9-3 所示，显示一个简易播放器。如果单击【播放】按钮，将开始播放音乐；如果单击【暂停】按钮，将暂停音乐的播放，此时按钮名称变为“继续”，若单击【继续】按钮，则可继续播放音乐；如果单击【停止】按钮，将停止音乐的播放，此时将无法再播放音乐。

【任务 9-3】 实现在线音乐播放

【任务描述】

编写程序制作一个简易在线音乐播放器，实现在线音乐播放功能，该音乐播放器有【播放网络音乐】、【暂停】和【停止】按钮，并且按钮下面有一个音乐播放的进度条，当单击对应按钮时实现对应操作。

音乐播放由 MediaPlayer 这个类控制，进度条 SeekBar 可以用来显示播放进度，用户也可以利用 SeekBar 的滑块来控制音乐的播放。界面中的两个按钮一个用来播放歌曲启动线程，另一个停止播放和取消线程。

【知识索引】

（1）SeekBar 类及 setMax()、setProgress()、setOnSeekBarChangeListener()、onProgressChanged()等方法。

（2）Runnable 类、Handler 类及 run()、post()、removeCallbacks()、postDelayed()等方法。

（3）MediaPlayer 类及 reset()、setDataSource()、prepare()、start()、stop()、release()、seekTo()等方法。

（4）try…catch 语句。

【实施过程】

1．创建 Android 项目 App0903

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0903。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为帧布局方式（FrameLayout），删除默认添加的 TextView 控件，然后在帧布局内添加 1 个线性布局，在外层线性布局内再添加 2 个线性布局。在第 1 个内层线性布局中添加 3 个 Button 控件，在第 2 个内层线性布局中添加 1 个 SeekBar 控件，并设置好各个控件的属性。布局文件 activity_main.xml 中控件对应的代码如表 9-9 所示。

表 9-9　布局文件 activity_main.xml 中控件对应的代码

序号	布 局 代 码
01	<FrameLayout …… >
02	<LinearLayout
03	……
04	android:layout_gravity="top"
05	android:orientation="vertical" >
06	<LinearLayout
07	……
08	android:layout_gravity="center_horizontal"
09	android:layout_marginTop="4.0dip"
10	android:orientation="horizontal" >
11	<Button
12	android:id="@+id/btnPlayUrl"

续表

序号	布 局 代 码
13	……
14	android:text="播放网络音频" >
15	</Button>
16	<Button
17	android:id="@+id/btnPause"
18	android:layout_width="80dip"
19	android:layout_height="wrap_content"
20	android:text="暂停" >
21	</Button>
22	<Button
23	android:id="@+id/btnStop"
24	android:layout_width="80dip"
25	android:layout_height="wrap_content"
26	android:text="停止" >
27	</Button>
28	</LinearLayout>
29	<LinearLayout
30	……
31	android:layout_marginBottom="20dip"
32	android:orientation="horizontal" >
33	<SeekBar
34	android:id="@+id/skbProgress"
35	……
36	android:layout_gravity="center_vertical"
37	android:layout_weight="1"
38	android:paddingLeft="10dip"
39	android:paddingRight="10dip">
40	</SeekBar>
41	</LinearLayout>
42	</LinearLayout>
43	</FrameLayout>

3．完善 MainActivity 类与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明多个对象，代码如下所示：

```
private boolean isPause = false;  //是否暂停
String url = "http://www.hnrpc.com:81/hntdsx/m01.mp3";  // MP3 的网络地址
private Button mBtnPlayUrl;       //播放
private Button mBtnPause;         //暂停
private Button mBtnStop;          //停止
private SeekBar mSkbProgress;     //进度条
private MediaPlayer player;
```

（2）在 onCreate()方法中编写代码实现程序功能

在 onCreate()方法中获取布局的控件对象，然后通过自定义方法 setListener()设置控件对象的监听器，其代码如表 9-10 所示。

表 9-10　onCreate()方法的代码

序号	代　　码
01	@Override
02	public void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	player = MediaPlayer.create(this, R.raw.m01);
06	mBtnPlayUrl = (Button) findViewById(R.id.btnPlayUrl);
07	mBtnPause = (Button) this.findViewById(R.id.btnPause);
08	mBtnStop = (Button) findViewById(R.id.btnStop);
09	mSkbProgress = (SeekBar) findViewById(R.id.skbProgress);
10	//获得歌曲的长度并设置成播放进度条的最大值
11	mSkbProgress.setMax(player.getDuration());
12	setListener();
13	}

（3）编写代码定义 Handler 对象和 Runnable 对象

定义 Handler 对象和 Runnable 对象的代码如表 9-11 所示，且重定义了 run()方法，该方法的功能分别是获取歌曲现在的播放位置并设置播放进度条的值、延迟 100 毫秒再启动线程。

表 9-11　定义 Handler 对象和 Runnable 对象的代码

序号	代　　码
01	Handler handler = new Handler();
02	Runnable updateThread = new Runnable() {
03	public void run() {
04	//获得歌曲现在播放位置并设置成播放进度条的值
05	mSkbProgress.setProgress(player.getCurrentPosition());
06	//每次延迟 100 毫秒再启动线程
07	handler.postDelayed(updateThread, 100);
08	}
09	};

（4）编写代码定义方法 playUrl

方法 playUrl()用于在线播放网络音乐，其代码如表 9-12 所示。

表 9-12　方法 playUrl()的代码

序号	代　　码
01	//在线播放音乐
02	public void playUrl(String videoUrl) {

续表

序号	布 局 代 码
03	try {
04	// 设备初始化
05	player.reset();
06	// 设置数据源
07	player.setDataSource(videoUrl);
08	// prepare 之后自动播放
09	player.prepare();
10	player.start();
11	} catch (IllegalArgumentException e) {
12	e.printStackTrace();
13	} catch (IllegalStateException e) {
14	e.printStackTrace();
15	} catch (IOException e) {
16	e.printStackTrace();
17	}
18	}

（5）编写代码为按钮自定义单击监听器

编写代码定义单击监听器 OnClickListener，方法 setListener()的实现代码如表 9-13 所示。

表 9-13　实现自定义单击监听器方法 setListener()的代码

序号	代　　码
01	private void setListener() {
02	mBtnPlayUrl.setOnClickListener(new Button.OnClickListener() {
03	@Override
04	public void onClick(View arg0) {
05	playUrl(url);
06	//启动
07	handler.post(updateThread);
08	if (isPause) {
09	mBtnPause.setText("暂停");
10	isPause = false;　　//设置暂停标记变量的值为 false
11	}
12	}
13	});
14	mBtnPause.setOnClickListener(new Button.OnClickListener() {
15	@Override
16	public void onClick(View arg0) {
17	player.pause();　　//音乐暂停
18	handler.post(updateThread);
19	mBtnPause.setText("继续");
20	if (!isPause) {

续表

序号	布 局 代 码
21	player.pause();　//暂停播放;
22	isPause = true;
23	mBtnPause.setText("继续");
24	} else {
25	playUrl(url);　//继续播放
26	mBtnPause.setText("暂停");
27	isPause = false;
28	}
29	}
30	});
31	mBtnStop.setOnClickListener(new Button.OnClickListener() {
32	@Override
33	public void onClick(View arg0) {
34	player.stop();
35	//取消线程
36	handler.removeCallbacks(updateThread);
37	player.release();
38	}
39	});
40	mSkbProgress.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
41	@Override
42	public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
43	// fromUser 判断是用户改变的滑块的值
44	if (fromUser == true) {
45	player.seekTo(progress);
46	}
47	}
48	});
49	}
50	

4．在 AndroidManifest.xml 文件中增加获得系统的网络状态权限

在 AndroidManifest.xml 文件中输入如下所示的代码，增加获得系统网络状态权限的代码，以保证程序可以获得系统的网络状态权限。

```
<uses-permission android:name="android.permission.INTERNET" />
```

5．程序运行与功能测试

Android 项目 App0903 的初始运行状态如图 9-4 所示。

单击【播放网络音频】按钮，开始播放音乐，进度条 SeekBar 也会跟着音乐播放的进度滑

块向右移动，如图 9-5 所示。当音乐播放完毕时，SeekBar 也会移动到最后。



图 9-4　Android 项目 App0903 的初始运行状态



图 9-5　播放音乐时进度条的滑块向右移动

【任务 9-4】使用 VideoView 控件播放本地视频

【任务描述】

编写程序创建一个简易视频播放器，使用 VideoView 实现本地视频的播放。

【知识索引】

（1）VideoView 类及 setMediaController()、setVideoURI()、start()、getCurrentPosition()、stopPlayback()、seekTo()等方法。

（2）MediaController 类、Uri 类及 parse()方法。

【实施过程】

1．创建 Android 项目 App0904

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0904。

在项目 App0904 的文件夹 res 中创建子文件夹“raw”，并将本任务所需的视频文件导入或复制到 res/ raw 文件夹中，且将所需的背景图片文件导入或复制到 drawable 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），删除默认添加的 TextView 控件。然后在线性布局管理器中添加一个 VideoView 控件，将 VideoView 控件的 id 属性设置为 videoview。

3．完善类 MainActivity 与实现程序功能

（1）设置 MainActivity 类需要实现的接口

类 MainActivity 的父类为 Activity，并实现了 MediaPlayer 类的 OnErrorListener 接口和 OnCompletionListener 接口，对应代码如下所示。

```
public class MainActivity extends Activity implements
MediaPlayer. OnErrorListener,
        MediaPlayer.OnCompletionListener {
      ……
}
```

（2）声明对象

在 MainActivity 类定义中，首先声明视频播放对象 VideoView 和视频控制播放对象 MediaController，然后声明 Uri 对象和记录当前播放位置的变量，代码如下所示：

```
private VideoView mVideoView;                    //定义视频播放对象
private MediaController mMediaController;       //定义视频控制播放对象
private Uri mUri;                                //定义资源定位对象
private int mPositionWhenPaused = -1;
```

（3）在 onCreate()方法中编写代码实现程序功能

onCreate()方法的代码如表 9-14 所示，首先获取了布局中的 VideoView 对象，然后创建音频控制对象、设置 MediaController 和视频文件路径。

表 9-14　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	//设置当前 Activity 的布局
05	setContentView(R.layout.activity_main);
06	mVideoView = (VideoView) findViewById(R.id.videoview);
07	// 创建音频控制对象
08	mMediaController = new MediaController(this);
09	// 设置 MediaController
10	mVideoView.setMediaController(mMediaController);
11	// 设置视频文件路径
12	mUri = Uri.parse("android.resource://" + getPackageName() + "/" +R.raw.v04);
13	}

（4）重写多个方法

重写 onError()方法实现监听 MediaPlayer 报告的错误信息，当视频播放出错时调用该方法；重写 onCompletion()方法实现 Video 播完的时候得到通知，当视频播放完毕时调用该方法；重写 onStart()方法实现视频开始播放，重写 onPause()方法实现暂停视频播放，重写 onResume()方法实现回放视频，当视频恢复播放时调用该方法。这些重写方法对应的代码如表 9-15 所示。

表 9-15　重写方法对应的代码

序号	代　　码
01	// 监听 MediaPlayer 报告的错误信息
02	@Override
03	public boolean onError(MediaPlayer mp, int what, int extra) {
04	return false;
05	}
06	// Video 播完的时候得到通知
07	@Override
08	public void onCompletion(MediaPlayer mp) {
09	this.finish();
10	}
11	//开始播放
12	public void onStart() {
13	//播放视频
14	mVideoView.setVideoURI(mUri);

续表

序号	布 局 代 码
15	mVideoView.start();
16	super.onStart();
17	}
18	//暂停播放
19	public void onPause() {
20	mPositionWhenPaused = mVideoView.getCurrentPosition();
21	mVideoView.stopPlayback();
22	super.onPause();
23	}
24	public void onResume() {
25	// 回放视频
26	if (mPositionWhenPaused >= 0) {
27	mVideoView.seekTo(mPositionWhenPaused);
28	mPositionWhenPaused = -1;
29	}
30	super.onResume();
31	}

4．在 AndroidManifest.xml 文件中增加相应的权限

在 AndroidManifest.xml 文件输入如下所示的代码，增加相应的权限。

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

5．程序运行与功能测试

Android 项目 App0904 的运行画面如图 9-6 所示。



图 9-6　Android 项目 App0904 的运行画面

【任务 9-5】使用 MediaPlayer 类和 SurfaceView 控件播放本地视频

【任务描述】

编写程序实现使用 MediaPlayer 类和 SurfaceView 控件组合播放本地视频，其中使用 MediaPlayer 类播放视频文件，使用 SurfaceView 控件显示视频图像。

【知识索引】

（1）MediaPlayer 类及 create()、setDisplay()、start()、isPlaying()、pause()、stop()、release()等方法。

（2）SurfaceView 类、getHolder()方法。

（3）SurfaceHolder 类、SURFACE_TYPE_PUSH_BUFFERS 常量、getHolder()方法、setType()方法。

【实施过程】

1．创建 Android 项目 App0905

在 Android Studio 集成开发环境中创建 Android 项目，将该项目命名为 App0905。

在项目 App0905 的文件夹 res 中创建子文件夹“raw”，并将本任务所需的视频文件导入或复制到 res/ raw 文件夹中，且将所需的背景图片文件导入或复制到 drawable 文件夹中。

2．完善布局文件 activity_main.xml 与界面设计

先将默认的相对布局方式（RelativeLayout）修改为线性布局方式（LinearLayout），删除默认添加的 TextView 控件。然后在线性布局管理器中添加一个 SurfaceView 控件（用于显示视频图像）和 1 个水平线性布局管理器 LinearLayout，并在该水平布局管理器中添加 3 个 Button 控件，分别为“播放”按钮、“暂停/继续”和“停止”按钮，并设置好各个控件的属性。布局文件 activity_main.xml 中控件对应的代码如表 9-16 所示。

表 9-16　布局文件 activity_main.xml 中控件对应的代码

序号	布 局 代 码
01	<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
02	android:layout_width="fill_parent"
03	android:layout_height="fill_parent"
04	android:gravity="center"
05	android:orientation="vertical" >
06	<SurfaceView
07	android:id="@+id/surfaceView1"
08	android:background="@drawable/bg"
09	android:keepScreenOn="true"
10	android:layout_width="576px"
11	android:layout_height="432px"/>
12	<LinearLayout
13	android:id="@+id/linearLayout1"
14	…… >
15	<Button
16	android:id="@+id/play"

续表

序号	布 局 代 码
17	……
18	android:text="播放" />
19	<Button
20	android:id="@+id/pause"
21	……
22	android:enabled="false"
23	android:text="暂停" />
24	<Button
25	android:id="@+id/stop"
26	……
27	android:text="停止" />
28	</LinearLayout>
29	</LinearLayout>

表 9-16 中第 06～11 行在布局管理器中定义了 1 个 SurfaceView 控件，其中第 08 行的 background 属性用于设置背景，第 09 行的 keepScreenOn 属性用于指定在播放视频时，是否打开屏幕。

3．完善类 MainActivity 与实现程序功能

（1）声明对象

在 MainActivity 类定义中，首先声明 MediaPlayer 对象、SurfaceView 对象和 SurfaceHolder 对象，代码如下所示：

```
private MediaPlayer mediaplayer;      //声明 MediaPlayer 对象
private SurfaceView sview;            //声明 SurfaceView 对象
private SurfaceHolder holder;         //声明 SurfaceHolder 对象
```

（2）在 onCreate()方法中编写代码实现程序功能

onCreate()方法的代码如表 9-17 示。第 06 行设置为横屏播放，第 07～09 行获取布局管理器中的按钮对象，第 10 行获取布局中添加的 SurfaceView 控件，第 11 行使用 create()方法实例化 MediaPlayer 对象。第 13～25 行为【播放】按钮添加单击事件监听器，第 27～38 行为【暂停】按钮添加单击事件监听器，第 40～49 行为【停止】按钮添加单击事件监听器，第 51～57 行为 MediaPlayer 对象添加完成事件监听器，并在重写的 onClick()方法中，实现播放、暂停/继续播放和停止播放视频等功能。

表 9-17　onCreate()方法的代码

序号	代　　码
01	@Override
02	protected void onCreate(Bundle savedInstanceState) {
03	super.onCreate(savedInstanceState);
04	setContentView(R.layout.activity_main);
05	//设置为横屏播放

续表

序号	布 局 代 码
06	this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
07	Button play=(Button)findViewById(R.id.play);　　//获取【播放】按钮
08	final Button pause=(Button)findViewById(R.id.pause);　//获取【暂停/继续】按钮
09	Button stop =(Button)findViewById(R.id.stop);　　//获取【停止】按钮
10	sview=(SurfaceView)findViewById(R.id.surfaceView);　//获取布局中添加的 SurfaceView 控件
11	mediaplayer=MediaPlayer.create(this,R.raw.mp01);
12	//为【播放】按钮添加单击事件监听器
13	play.setOnClickListener(new OnClickListener() {
14	@Override
15	public void onClick(View v) {
16	//mediaplayer.setDataSource("/sdcard/mp01.mp4");　//设置要播放的视频
17	//mediaplayer.setDisplay(sview.getHolder());　//设置将视频画面输出到 SurfaceView
18	holder = sview.getHolder();
19	holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
20	mediaplayer.setDisplay(holder);
21	mediaplayer.start();　//开始播放
22	pause.setText("暂停");
23	pause.setEnabled(true);　//设置【暂停】按钮可用
24	}
25	});
26	//为【暂停】按钮添加单击事件监听器
27	pause.setOnClickListener(new OnClickListener() {
28	@Override
29	public void onClick(View v) {
30	if(mediaplayer.isPlaying()){
31	mediaplayer.pause();　　//暂停视频的播放
32	((Button)v).setText("继续");
33	}else{
34	mediaplayer.start();　　//继续视频的播放
35	((Button)v).setText("暂停");
36	}
37	}
38	});
39	//为【停止】按钮添加单击事件监听器
40	stop.setOnClickListener(new OnClickListener() {
41	@Override
42	public void onClick(View v) {
43	if(mediaplayer.isPlaying()){
44	mediaplayer.stop();　　//停止播放
45	sview.setBackgroundResource(R.drawable.bg_finish); //改变 SurfaceView 的背景图片
46	pause.setEnabled(false);　//设置【暂停】按钮不可用
47	}

续表

序号	布 局 代 码
48	} });
49	//为 MediaPlayer 对象添加完成事件监听器
50	mediaplayer.setOnCompletionListener(new OnCompletionListener() {
51	@Override
52	public void onCompletion(MediaPlayer mp) {
53	sview.setBackgroundResource(R.drawable.bg_finish); //改变 SurfaceView 的背景图片
54	Toast.makeText(MainActivity.this, "视频播放完毕！", Toast.LENGTH_SHORT).show();
55	}
56	});
57	}
58	

（3）重写 Activity 的 onDestroy()方法

重写 Activity 的 onDestroy()方法，其代码如表 9-18 所示，用于在当前 Activity 销毁时，停止正在播放的视频，并释放 MediaPlayer 所占用的资源。

表 9-18　onDestroy()方法的代码

序号	代　　码
01	@Override
02	protected void onDestroy() {
03	if(mediaplayer.isPlaying()){
04	mediaplayer.stop(); //停止播放视频
05	}
06	mediaplayer.release(); //释放资源
07	super.onDestroy();
08	}

4．程序运行与功能测试

Android 项目 App0905 的初始运行状态如图 9-7 所示。单击【播放】按钮，将开始播放视频，播放画面如图 9-8 所示，并且让【暂停】按钮可用；单击【暂停】按钮，将暂停视频的播放，同时该按钮变为【继续】按钮；单击【停止】按钮，将停止正在播放的视频。



图 9-7　Android 项目 App0905 的初始运行状态



图 9-8　App0905 播放视频的画面

视频播放完毕时显示如图 9-9 所示的画面。



图 9-9　视频播放完毕时显示的画面

【说明】：由于这里采用的是模拟器运行，所以视频的画面并没有显示，视频的声音可以听到，而在屏幕中间显示的图片是为 SurfaceView 控件设置的背景图片。如果将该程序发布到真机中运行，可以看到视频画面。

单元小结

Android 系统能够录制、播放各种不同形式的多媒体文件，Android 的多媒体系统为 Android 多媒体应用开发提供了非常好的平台。Android 平台中播放音频主要有以下两种方式：SoundPool 适合于短促且对反应速度比较高的情况（如游戏音效或按键声等），MediaPlayer 适合于比较长且对时间要求不高的情况。在 Android 中，播放视频主要使用 VideoView 或者 SurfaceView，其中 VideoView 组件播放视频最简单。

本单元主要介绍了 Android 中播放声音的主要类 SoundPool 和 MediaPlayer、播放视频的主要类 VideoView 和 SurfaceView。通过完成多个任务学会了使用 SoundPool 类播放音频、使用 MediaPlayer 类播放本地音频和网络音频、使用 VideoView 控件播放本地视频、使用 MediaPlayer 类和 SurfaceView 控件播放本地视频。

单元习题

一、填空题

1．Android 中开发音乐播放器可以使用（　　）类和（　　）类。

2．Android 中开发开发视频播放器可以使用（　　）或者（　　），其中（　　）组件播放视频最简单，它将视频的显示和控制集于一身，因此，借助它就可以完成一个简易的视频播放器。

3．SurfaceView 继承自（　　）类，是用于显示图像的组件，其最大特点就是它的（　　）技术。

4．为 MediaPlayer 指定加载的音频文件时可以使用 MediaPlayer 提供的静态方法（　　）和非静态方法（　　）。

5．调用 prepareAsync()方法会使 MediaPlayer 对象进入（　　）状态。

二、选择题

1．MediaPlayer 播放资源时，需要调用（　　）方法完成准备工作。

A．setDataSource　　B．prepare　　C．begin　　D．pause

2．Android 中 MediaPlayer 无法播放（　　）。

A．程序资源文件　　B．网络上的文件

C．SD 卡上的文件　　D．其他程序资源文件

3．MediaPlayer 对象执行（　　）之后处于 Started 状态。

A．start()　　B．stop()　　C．pause()　　D．reset()

4．调用 stop()方法会停止 MediaPlayer 对象播放，并且还会让一个处于 Started，Paused，Prepared 或 PlaybackCompleted 状态的 MediaPlayer 进入（　　）状态。

A．Started　　B．Stopped　　C．Paused　　D．Prepared

5．使用 MediaPlayer 播放保存在 SD 卡中的 MP3 文件时（　　）。（多选题）

A．需要使用 MediaPlayer.create 方法创建 MediaPlayer

B．直接使用 new MediaPlayer 即可

C．直接使用 setDataSource 方法设置文件源即可

D．直接调用 start 方法，无须设置文件源

三、简答题

1．简要说明使用 MediaPlayer 播放音频时，设置音频数据源的主要方法有哪些。

2．简要说明 MediaPlayer 对象的生命周期。

3．简述 MediaPlayer 对象的 prepareAsync()方法和 prepare()方法的主要区别。

4．简要说明使用 VideoView 类播放视频的主要步骤。

附录 A

Android 移动应用开发课程设计

1．教学单元设计

单元序号	单元名称	任务数量	建议课时	建议考核分值
单元 1	Android 开发环境搭建与模拟器创建	2	4	5
单元 2	Android Studio 的组成结构与基本操作	4	6	5
单元 3	Android 的界面设计与资源应用程序设计	9	10	15
单元 4	Android 的事件处理与交互实现程序设计	6	8	15
单元 5	Android 的数据存储与数据共享程序设计	6	8	15
单元 6	Android 的服务与广播应用程序设计	6	8	15
单元 7	Android 的网络与通信应用程序设计	5	6	10
单元 8	Android 的图像浏览与图形绘制程序设计	5	6	10
单元 9	Android 的音频与视频应用程序设计	5	8	10
合计		48	64	100

2．教学流程设计

教学环节序号	教学环节名称	说　明
1	教学导航	明确教学目标、熟悉教学方法、了解课时建议
2	知识导读	讲解 Android 程序开发所涉及的概念、理论和方法，提供理论指导和方法支持
3	任务实战	设置了“任务描述→知识索引→实施过程”3 个环节，通过“知识索引”将各项任务所应用的知识与“知识导读”环节的理论知识关联起来，“实施过程”详细说明任务实现的全过程
4	单元总结	对单元教学内容和技能训练情况进行归纳总结，形成整体印象

3．操作任务设计

序　号	操作任务索引	
单元 1	【任务 1-1】搭建 Android 开发环境与创建模拟器	【任务 1-2】启动 Android Studio 与创建运行 Android Studio 项目
单元 2	【任务 2-1】熟悉 Android Studio 的组成结构 【任务 2-2】Android Studio 项目中的模块操作	【任务 2-3】Android Studio 开发环境的个性化设置 【任务 2-4】将 Android Studio 项目打包生成 APK

续表

序　　号	操作任务索引	
单元 3	【任务 3-1】使用文字标签显示欢迎信息 【任务 3-2】设计包含多种控件的用户登录界面 【任务 3-3】设计开关与调节声音的界面 【任务 3-4】使用 DatePicker 控件获取日期 【任务 3-5】使用日期时间控件获取与显示时间	【任务 3-6】使用 LinearLayout 布局设计用户注册界面 【任务 3-7】使用 TableLayout 布局设计商品信息展示界面 【任务 3-8】使用 GridLayout 布局设计仿 Windows 8 桌面 【任务 3-9】使用 FrameLayout 布局实现图片相框效果
单元 4	【任务 4-1】用户触摸屏幕时显示提示信息 【任务 4-2】用户登录时检测用户名的长度合法性 【任务 4-3】获取屏幕单击位置	【任务 4-4】实现动态添加联系人 【任务 4-5】用户注册时检测 Email 格式 【任务 4-6】打开浏览器浏览网页
单元 5	【任务 5-1】设计可记住用户名和密码的登录界面 【任务 5-2】对 Android 模拟器中的 SD 卡进行操作 【任务 5-3】使用 SharedPreferences 实现 Activity 之间的数据传递	【任务 5-4】使用 SQLite 数据库保存用户输入的用户名和密码 【任务 5-5】预览选择的系统图片 【任务 5-6】实现添加与查询联系人
单元 6	【任务 6-1】获取系统的唤醒服务 【任务 6-2】获取系统的屏蔽状态 【任务 6-3】获取当前网络状态	【任务 6-4】实现音量控制 【任务 6-5】实现程序开机自动启动 【任务 6-6】监控手机电池电量
单元 7	【任务 7-1】获取指定城市的天气预报 【任务 7-2】实现百度在线搜索 【任务 7-3】实现浏览网络图片	【任务 7-4】实现短信发送 【任务 7-5】实现电话拨打
单元 8	【任务 8-1】设计按钮切换的图片浏览器 【任务 8-2】设计滑动切换的图片浏览器 【任务 8-3】绘制简单几何图形	【任务 8-4】绘制多种形式的路径 【任务 8-5】绘制 Android 机器人图形
单元 9	【任务 9-1】使用 SoundPool 类播放音频 【任务 9-2】使用 MediaPlayer 类播放本地音频 【任务 9-3】实现在线音乐播放	【任务 9-4】使用 VideoView 控件播放本地视频 【任务 9-5】使用 MediaPlayer 类和 SurfaceView 控件播放本地视频
累计	48 项任务	

附录 B

Android 移动应用开发岗位需求调研

Android 移动应用开发职业岗位的工作职责和职位要求调研如下表所示。

企业名称	上海文和信息技术有限公司	职位名称	Android 开发工程师
岗位职责	职位要求		
（1）负责公司 Android 平台软件的开发工作 （2）与产品和 UI 人员一起完成产品的发布，保证产品的开发进度 （3）对客户端软件进行维护和持续升级；并根据产品策略对功能进行调整 （4）独立完成 Android 软件代码编写、验证和修正测试中发现的问题 （5）学习和研究新的 Android 平台技术和无线互联网新技术以满足产品功能需求	（1）熟悉 Java 开发语言，掌握 Java 线程机制，熟悉 Android 的 Handler 机制 （2）熟悉 Android 系统权限和 Android 四大组件 （3）熟悉 Android 系统 UI 框架，具有 UI 搭建能力 （4）熟悉 Android 系统网络编程，熟练掌握 HTTP、Socket 编程 （5）熟练使用 Android 程序的集成开发平台，熟悉 Android SDK、NDK、DEBUG 调试 （6）熟悉常用的如 XML、JSON 等数据封装与解析技术，了解常用的数据安全加密、PUSH 推送等相关技术 （7）具备良好的编码习惯，较强的文档处理能力，有较强的学习能力、解决问题的能力以及一定的项目压力适应能力 （8）工作认真负责、人品正直，具有良好的沟通能力与团队合作精神 （9）有 Android 应用商店上架作品或相关作品优先考虑 （10）熟悉智能卡、网络通信、手机支付等相关领域业务者优先		
企业名称	北京璧合科技有限公司	职位名称	Android 工程师
岗位职责	职位要求		
（1）根据产品需求开发 Android 客户端产品 （2）验证和修正测试中发现的问题 （3）学习和研究新技术以满足产品的需求 （4）根据开发过程中的体验对产品提出改进建议	（1）精通 Java 语言，有一定的 Android 开发经验 （2）精通 Android 开发平台及框架原理，精通 Android 基本组件的使用 （3）熟悉 Android 网络编程、网络数据传输和 JSON 数据传输 （4）熟悉 Android 的安全技术 （5）熟悉 Android 的多媒体技术、音频、视频、图片等处理 （6）熟练使用多种常用设计模式，具扎实的编程功底和良好的编程习惯，良好的设计文档撰写能力；能很好地高保真输出设计师的 UI （7）责任心强，良好的对外沟通和团队协作能力，能承受工作压力		

续表

企业名称	广州翼锋信息科技有限公司	职位名称	Android 软件工程师
岗位职责	职位要求		
（1）负责 Android 平台应用客户端程序的开发，包括 Android Phone，Android Pad （2）负责手机端应用市场类产品的设计和研发 （3）负责与服务端接口的交互和产品 UI 的开发 （4）承担并独立完成开发任务，并对交付质量负责 （5）完成 Android 应用架构的设计和规范化工作 （6）负责相关技术文档的编写	（1）熟悉 Android 开发平台的框架原理，精通 Java 编程技术 （2）能够熟练使用 Android SDK，熟练掌握 Android 网络应用软件开发和调试 （3）精通 Android 界面 UI 布局设计开发，包括多种分辨率，尺寸和栏目动态展现 （4）熟练掌握 JSON 和 XML 数据解析、HTTP 协议、Android 系统 API 调用，了解 REST 架构风格 （5）深刻理解面向对象的理念和设计模式，熟练掌握数据结构、基本算法与重构 （6）具有严谨的逻辑分析能力，代码写作规范，性能良好 （7）具备良好的团队合作精神，较强的文档处理能力和沟通能力，较强的责任心，具有创业心态和积极向上的精神 （7）具有手机应用市场开发经验者优先		
企业名称	山东速联信息科技有限公司	职位名称	Android 开发工程师
岗位职责	职位要求		
（1）负责搭建开发环境，参与需求分析和架构设计讨论 （2）从事 Android 平台手机终端软件设计、调试、维护与优化 （3）独立完成功能模块开发 （4）撰写相关开发文档 （5）参与产品单元测试，后期产品升级维护 （6）负责用户体验优化完善，追求卓越产品的创造和设计	（1）熟练掌握 Java 语言，能独立进行 Android 的应用开发 （2）熟悉 Android 应用开发框架，深入了解常用 Android 平台 SDK，了解 Android 的常用功能实现 （3）对 Android 平台的网络通信、多线程，内存机制有一定的了解；能开发出高性能的 Android 应用 （4）了解 Android APP 的主流界面和界面效果实现方法，能够根据产品文档及 UI 设计图完成 Android App 的开发与调试工作 （5）熟悉规范文档的编写与维护 （6）熟悉 Android 下的网络通信机制，熟悉 HTTP、TCP/IP、Socket 等网络通信编程技术 （7）有较强的学习能力、沟通能力、问题解决能力、组织协调能力		
企业名称	江苏智通交通科技有限公司	职位名称	Android 移动开发工程师
岗位职责	职位要求		
（1）负责 Android App 产品的设计、开发工作 （2）根据项目计划按时完成软件编码和单元测试工作 （3）按照开发流程编写相应模块的设计文档 （4）解决研发过程中的关键问题和技术难题 （5）协调和指导其他开发人员的开发工作 （6）协助开发经理保证研发工作的质量和进度	（1）熟悉 Android 框架及各种特性，有 Android 项目设计经验 （2）负责 Android 平台软件产品的核心模块的设计与开发，负责关键技术的突破 （3）参与到产品设计过程中，分析需求、把需求转化为产品； （4）精通 HTTP/JSON 和 XML 等网络协议，熟悉多线程及面向对象编程，熟悉 SQLLite 数据库编程 （5）熟练掌握 Java 进行手机平台上的编程开发 （6）熟练编写单元测试，了解自动化测试 （7）按时完成高质量的代码，注重代码细节 （8）思路清晰，思维敏捷，快速的学习能力，良好的英文资料阅读能力 （9）有良好的计划、沟通、组织协调能力、语言表达能力；较强的逻辑分析能力和应变能力；良好的团队合作精神，能承担较大工作压力		

附录 C 各单元任务中类及引入包的说明

本书各单元任务中相关类及引入相关包的代码如下表所示。

序号	类　名	引入相关的包
01	Activity	import android.app.Activity;
02	Bundle	import android.os.Bundle;
03	Menu	import android.view.Menu;
04	MenuItem	import android.view.MenuItem;
05	View	import android.view.View;
06	TextView	import android.widget.TextView;
07	EditText	import android.widget.EditText;
08	Button	import android.widget.Button;
09	ImageView	import android.widget.ImageView;
10	CheckBox	import android.widget.CheckBox;
11	MotionEvent	import android.view.MotionEvent;
12	KeyEvent	import android.view.KeyEvent;
13	Toas	import android.widget.Toast;
14	Gravity	import android.view.Gravity;
15	LayoutInflater	import android.view.LayoutInflater;
16	OnClickListener	import android.view.View.OnClickListener;
17	Intent	import android.content.Intent;
18	ContentResolver	import android.content.ContentResolver;
19	Bitmap	import android.graphics.Bitmap
20	BitmapFactory	import android.graphics.BitmapFactory;
21	Log	import android.util.Log;
22	FileNotFoundException	import java.io.FileNotFoundException;
23	Contacts	import android.provider.Contacts;
24	SharedPreferences	import android.content.SharedPreferences;
25	Editor	import android.content.SharedPreferences.Editor;
26	Environment	import android.os.Environment;
27	File	import java.io.File;
28	IOException	import java.io.IOException;
29	FileOutputStream	import java.io.FileOutputStream;
30	Cursor	import android.database.Cursor;
31	MediaStore	import android.provider.MediaStore;
32	ContactsContract	import android.provider.ContactsContract;
33	Context	import android.content.Context;

续表

序号	类　名	引入相关的包
34	PowerManager	import android.os.PowerManager;
35	WakeLock	import android.os.PowerManager.WakeLock;
36	KeyguardManager	import android.app.KeyguardManager;
37	ConnectivityManager	import android.net.ConnectivityManager;
38	NetworkInfo	import android.net.NetworkInfo;
39	State	import android.net.NetworkInfo.State;
40	AudioManager	import android.media.AudioManager;
41	IntentFilter	import android.content.IntentFilter;
42	WebChromeClient	import android.webkit.WebChromeClient;
43	WebView	import android.webkit.WebView;
44	WebViewClient	import android.webkit.WebViewClient;
45	BufferedReader	import java.io.BufferedReader;
46	URI	import java.net.URI;
47	HttpResponse	import org.apache.http.HttpResponse;
48	HttpClient	import org.apache.http.client.HttpClient;
49	HttpGet	import org.apache.http.client.methods.HttpGet;
50	EntityUtils	import org.apache.http.util.EntityUtils;
51	Handler	import android.os.Handler;
52	Message	import android.os.Message;
53	ProgressBar	import android.widget.ProgressBar;
54	InputStream	import java.io.InputStream;
55	HttpURLConnection	import java.net.HttpURLConnection;
56	URL	import java.net.URL;
57	Bitmap	import android.graphics.Bitmap;
58	TextUtils	import android.text.TextUtils;
59	SmsManager	import android.telephony.SmsManager;
60	LayoutParams	import android.view.ViewGroup.LayoutParams;
61	ImageSwitcher	import android.widget.ImageSwitcher;
62	GestureDetector	import android.view.GestureDetector;
63	Canvas	import android.graphics.Canvas;
64	RectF	import android.graphics.RectF;
65	Color	import android.graphics.Color;
66	Paint	import android.graphics.Paint;
67	Style	import android.graphics.Paint.Style;
68	FrameLayout	import android.widget.FrameLayout;
69	Path	import android.graphics.Path;
70	SoundPool	import android.media.SoundPool;
71	HashMap	import java.util.HashMap;
72	MediaPlayer	import android.media.MediaPlayer;
73	SeekBar	import android.widget.SeekBar;
74	MediaController	import android.widget.MediaController;

续表

序号	类　　名	引入相关的包
75	VideoView	import android.widget.VideoView;
76	SurfaceView	iimport android.view.SurfaceView;
77	SurfaceHolder	import android.view.SurfaceHolder;
78	ActivityInfo	import android.content.pm.ActivityInfo;

附录 D

下载、安装与配置 JDK

1．下载 JDK

（1）打开浏览器，输入网址 http://www.oracle.com/index.html，进入 Oracle 官网主页。

（2）在 Oracle 官网主页中选择【Downloads】菜单下的“Java for Developers”子菜单，如图 D-1 所示。

（3）在跳转的页面中找到下载 Java for Developers 的超链接，如图 D-2 所示。

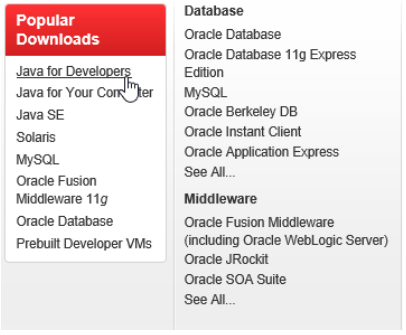


图 D-1　在 Downloads 菜单下选择子菜单“Java for Developers”

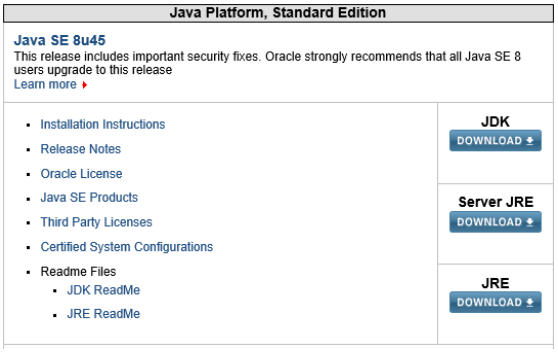
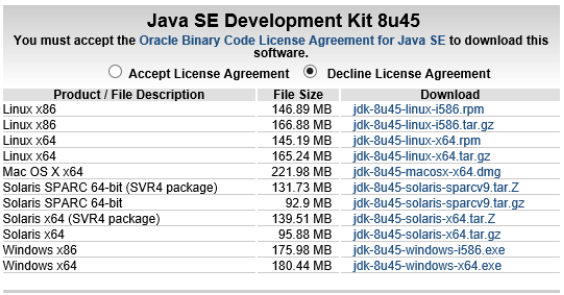


图 D-2　下载 Java for Developers 的超链接

（4）单击 JDK 下方的【DOWNLOAD】按钮，将进入如图 D-3 所示的 Java SE Development Kit 8u45 下载页面。

（5）在 Java SE Development Kit 8u45 下载页面中选中“Accept License Agreement”单选按钮，接受许可协议，并根据计算机硬件和操作系统版本选择适当的 JDK 版本进行下载。如果操作系统为 Windows 32 位，则下载 jdk-8u45-windows-i586.exe，如果操作系统是 Windows 64 位，则下载 jdk-8u45-windows-x64.exe。



Product / File Description	File Size	Download
Linux x86	146.89 MB	jdk-8u45-linux-i586.rpm
Linux x86	166.88 MB	jdk-8u45-linux-i586.tar.gz
Linux x64	145.19 MB	jdk-8u45-linux-x64.rpm
Linux x64	165.24 MB	jdk-8u45-linux-x64.tar.gz
Mac OS X x64	221.98 MB	jdk-8u45-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	131.73 MB	jdk-8u45-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	92.9 MB	jdk-8u45-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	139.51 MB	jdk-8u45-solaris-x64.tar.Z
Solaris x64	95.88 MB	jdk-8u45-solaris-x64.tar.gz
Windows x86	175.98 MB	jdk-8u45-windows-i586.exe
Windows x64	180.44 MB	jdk-8u45-windows-x64.exe

图 D-3　“Java SE Development Kit 8u45”下载列表

2．安装 JDK

双击 JDK 安装包 jdk-8u25-windows-x64-8.0.250.18.1413446274.exe，启动 Java SE Development Kit 8 Update 25(64-bit)安装向导，如图 D-4 所示。

单击【下一步】按钮，打开“定制安装”对话框，如图 D-5 所示。



图 D-4　安装 JDK 的欢迎界面



图 D-5　JDK 安装向导的“定制安装”对话框

单击【下一步】按钮，打开“安装进度”对话框，开始提取安装程序，如图 D-6 所示。

安装程序提取完毕，打开“目标文件夹”对话框，如图 D-7 所示，如需更改目标文件夹则单击【更改】按钮，打开一个对话框，重新选择目标文件夹即可，这里不更改目标文件夹，直接单击【下一步】按钮即可。

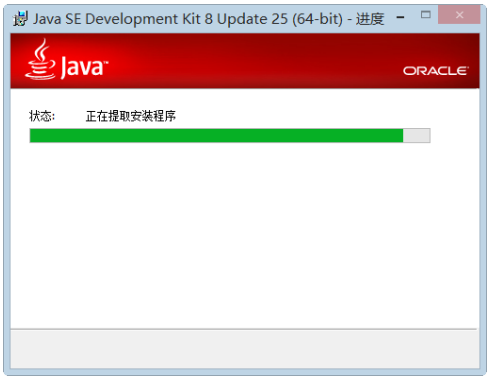


图 D-6　JDK 安装向导的“安装进度”对话框



图 D-7　JDK 安装向导的【目标文件夹】对话框

单击【下一步】按钮，开始安装 Java，如图 D-8 所示。



图 D-8　Java 安装进度界面

Java SE Development Kit 8 Update 25(64-bit)成功安装完成，打开如图 D-9 所示的对话框，单击【关闭】按钮即可。

3．配置 JDK

在 Windows 操作系统中打开如图 D-10 所示的“系统属性”对话框。



图 D-9　JDK 安装完成界面

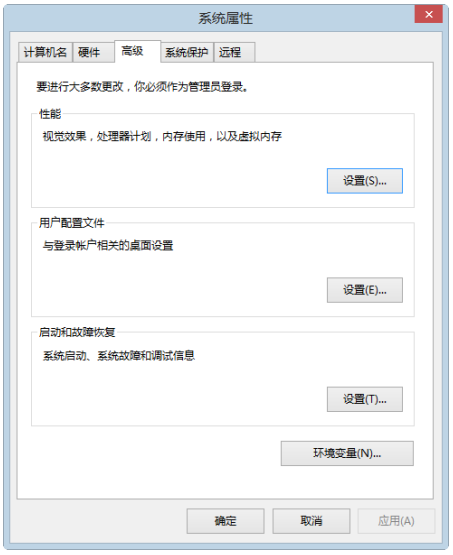


图 D-10　“系统属性”对话框

（1）编辑系统变量“Path”

在“系统属性”对话框中单击【环境变量】按钮，打开“环境变量”对话框，在下方的环境变量区域选择已有变量“Path”，单击【编辑】按钮，打开“编辑系统变量”对话框，将光标置于“变量值”文本框中，然后按【Home】键，输入“C:\Program Files\Java\jdk1.8.0_25\bin;”，如图 D-11 所示。单击【确定】按钮，返回“环境变量”对话框，如图 D-12 所示。

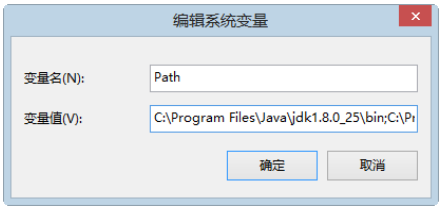


图 D-11　“编辑系统变量”对话框

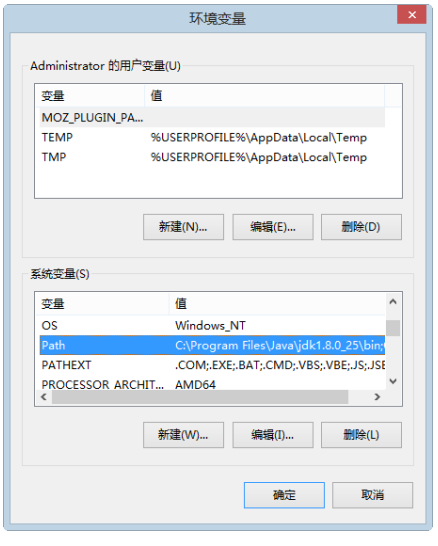
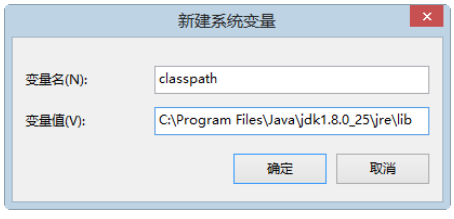


图 D-12　系统变量“Path”编辑完成后的“环境变量”对话框

（2）新建系统变量“classpath”

在“环境变量”对话框中单击“系统变量”区域的【新建】按钮，打开“新建系统变量”对话框，在该对话框的“变量名”文本框中输入“classpath”，在“变量值”文本框中输入：“C:\Program Files\Java\jdk1.8.0_25\jre\lib”（注意，这里根据实际安装路径填写），如图 D-13 所示。

单击【确定】按钮，返回“环境变量”对话框，如图 D-14 所示。



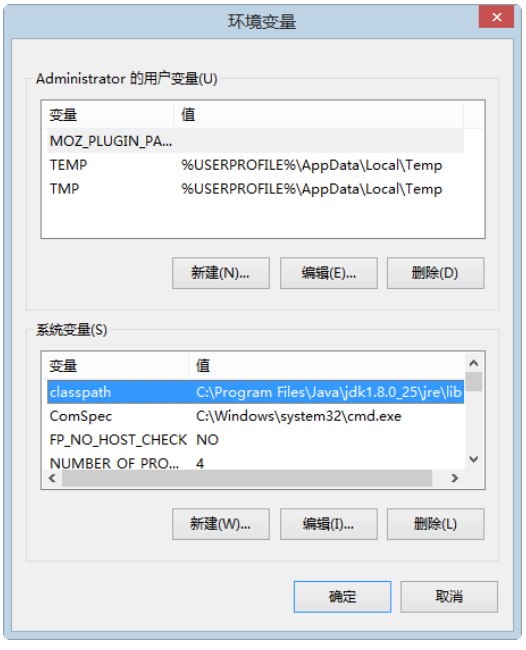


图 D-13　【新建系统变量】对话框　图 D-14　系统变量“classpath”新建完成后的“环境变量”对话框

在“环境变量”对话框中单击【确定】按钮，完成环境变量的配置。

说明：在“环境变量”对话框中先查看是否有 classpath 变量，如果已经存在就选中 classpath 选项，单击【编辑】按钮，然后在弹出的“编辑系统变量”对话框中进行编辑操作。否则就单击【新建】按钮新建一个环境变量 classpath。

（3）新建用户变量“JAVA_HOME”

在“环境变量”对话框中单击“用户变量”区域的【新建】按钮，打开“新建用户变量”对话框，在该对话框的“变量名”文本框中输入“JAVA_HOME”，在“变量值”文本框中输入：“C:\Program Files\Java\jdk1.8.0_25”（注意，这里根据实际安装路径填写），如图 D-15 所示。

接着在“新建用户变量”对话框中单击【确定】按钮返回“环境变量”对话框，如图 D-16 所示。

注意：这里必须新建用户变量“JAVA_HOME”，否则在编译与运行 Android 程序时会弹出如图 D-17 所示的提示信息对话框。

JDK 的环境变量配置完成后，可能通过打开“命令提示符”窗口输入命令“java -version”进行测试，从显示的 Java 版本的信息来确定安装是否成功，如图 D-18 所示。

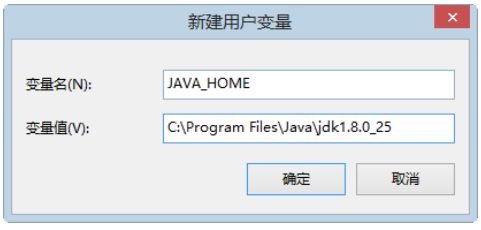


图 D-15　“新建用户变量”对话框

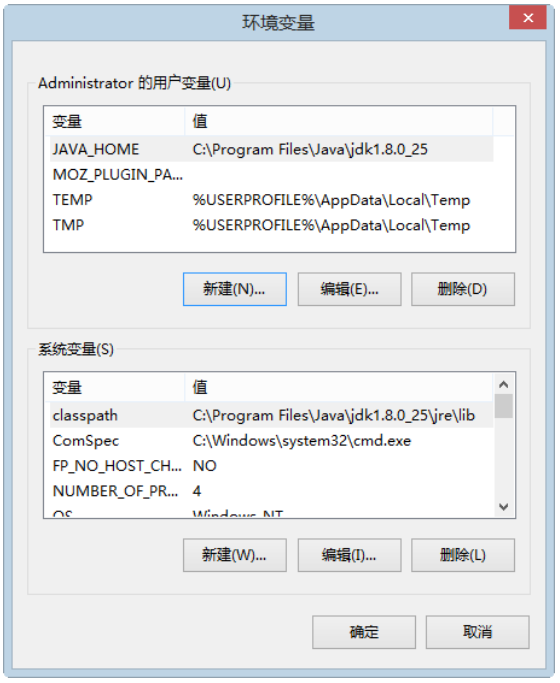


图 D-16　用户变量“JAVA_HOME”新建完成后的“环境变量”对话框

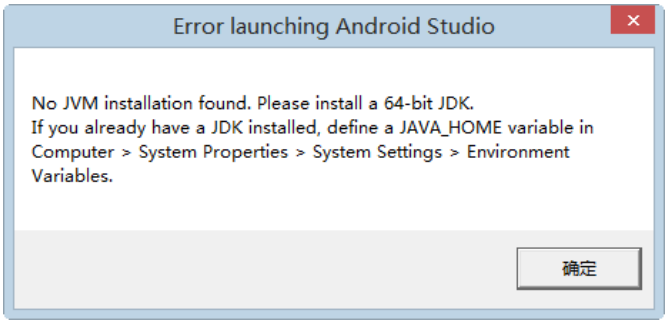


图 D-17　“Error launching Android Studio”提示信息对话框

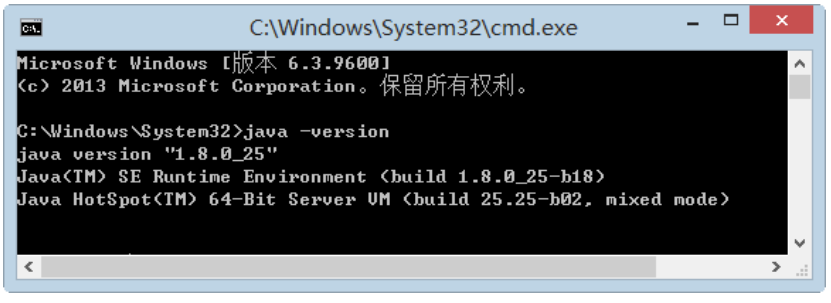


图 D-18　验证 JDK 安装是否成功

附录 E 下载与安装 Android Studio

1．下载 Android Studio

（1）打开浏览器，输入网址 http://www.android-studio.org/，进入 Android Studio 中文社区的主页。

（2）在 Android Studio 中文社区的主页选择最新版本的 Android Studio，这里选择的版本为 Android Studio V1.2.1，如图 E-1 所示。



图 E-1　Android Studio 中文社区的主页

然后下载 Android Studio V1.2.1。

2．安装 Android Studio

双击安装文件 android-studio-bundle-135.1629389.exe，启动 Android Studio 安装向导，首先显示如图 E-2 所示的“Android Studio Setup”欢迎界面。

单击【Next】按钮，进入“Choose Components”界面，如图 E-3 所示。该界面第 1 个选项“Android Studio 程序”为必选项；第 2 个选项“Android SDK”如果你的电脑中已经存在 Android SDK 可以不勾选；第 3 个选项和第 4 个选项都和虚拟器有关系，如果不使用虚拟器或者 SDK 中的虚拟器，可以不勾选。

单击【Next】按钮，进入“License Agreement”界面，如图 E-4 所示。

单击【I Agree】按钮，进入“Configuration Settings”界面，如图 E-5 所示。

在“Configuration Settings”界面，设置或输入 Android Studio 和 Android SDK 的安装文件夹。然后单击【Next】按钮，如果在前面的安装步骤中安装了 Intel HAXM，则会出现如图 E-6 所示的设置内存大小的界面，这里选择默认值“512MB”即可。



图 E-2 “Android Studio Setup”欢迎界面

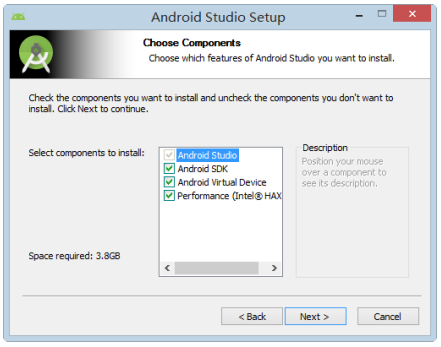


图 E-3 “Android Studio Setup”的“Choose Components”界面

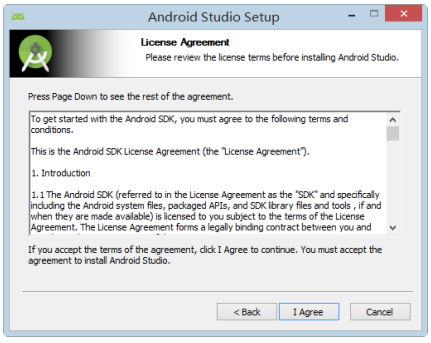


图 E-4 “Android Studio Setup”的“License Agreement”界面

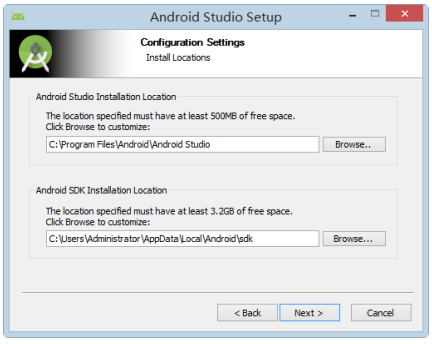


图 E-5 【Android Studio Setup】的“Configuration Settings”界面

单击【Next】按钮，进入修复 Microsoft Visual C++的界面，如图 E-7 所示，由于 Android Studio 的运行需要 **VC++**环境，Android Studio 安装的过程中，会自动安装，这也是为什么建议使用安装包（exe）的原因。

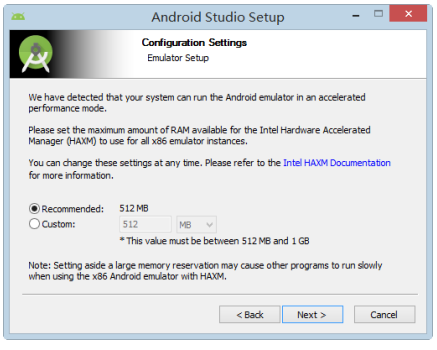


图 E-6 设置内存大小界面

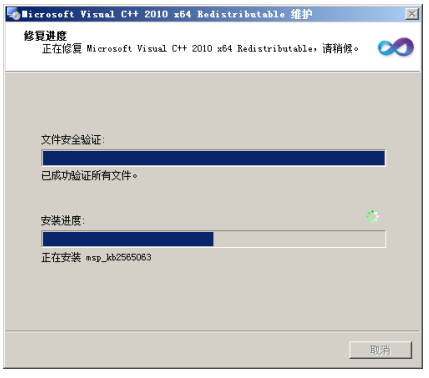


图 E-7 修复 Microsoft Visual C++界面

Microsoft Visual C++修复完成后，进入“Choose Start Menu Folder”界面设置开始菜单位置和名称，如图 E-8 所示，这里采用默认设置。

单击【Install】按钮，开始安装 Android Studio Setup，安装进度界面如图 E-9 所示。

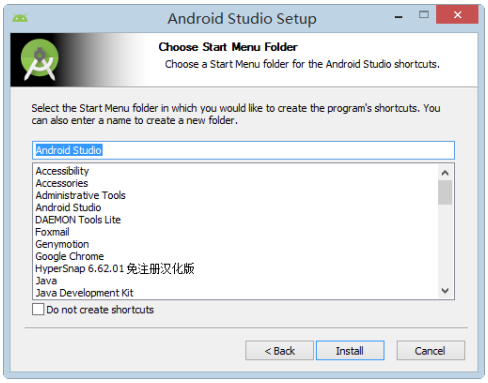


图 E-8　“Android Studio Setup”的“Choose Start Menu Folder”界面

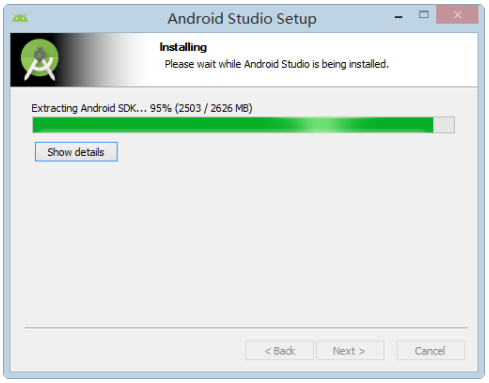


图 E-9　“Android Studio Setup”的“Installing”界面

Android Studio 成功安装完成后，单击【Show Details】按钮，显示其安装详情，如图 E-10 所示。

单击【Next】按钮，进入“Completing Android Studio Setup”界面，如图 E-11 所示，单击【Finish】按钮完成 Android Studio 的安装。

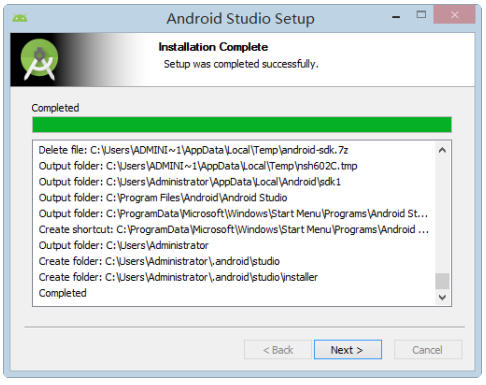


图 E-10　“Android Studio Setup”的“Installation Complete”界面

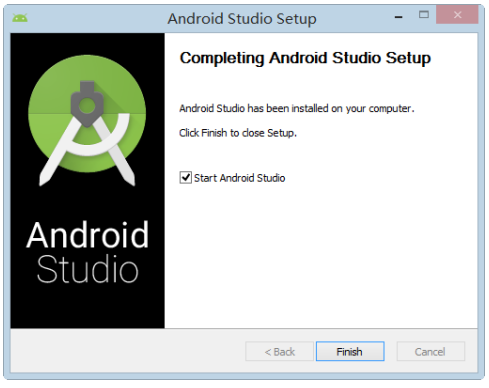


图 E-11　“Android Studio Setup”的“Completing Android Studio Setup”界面

附录 F 下载与安装英特尔硬件加速执行管理器

1．下载英特尔硬件加速执行管理器

（1）打开浏览器，输入网址 https://software.intel.com/zh-cn/android/articles/intel-hardware-accelerated-execution-manager/，进入英特尔硬件加速执行管理器的下载页面。

（2）在下载页面单击下载链接“haxm-windows_r05.zip (1.1.1)”，即可开始下载。

2．安装英特尔硬件加速执行管理器

haxm-windows_r05.zip 下载完毕后进行解压缩，然后双击 intelhaxm.exe，启动安装程序，打开如图 F-1 所示的英特尔硬件加速执行管理器启动界面。



图 F-1　英特尔硬件加速执行管理器的启动界面

单击【Next】按钮，进入内存配置界面，如图 F-2 所示，在该界面中可以对分配给英特尔硬件加速执行管理器的 RAM 量进行调整。

单击【Next】按钮，进入如图 F-3 所示内存大小等提示信息界面。

单击【Install】按钮即开始安装程序，如图 F-4 所示。

英特尔硬件加速执行管理器安装完成后，单击【Finish】按钮退出安装程序，如图 F-5 所示。

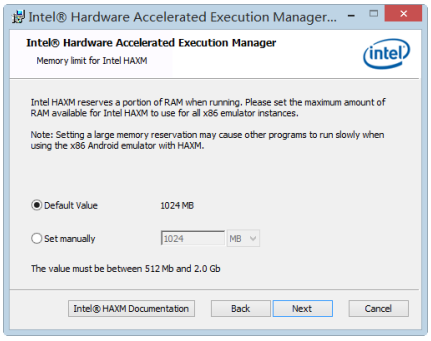


图 F-2　内存配置界面

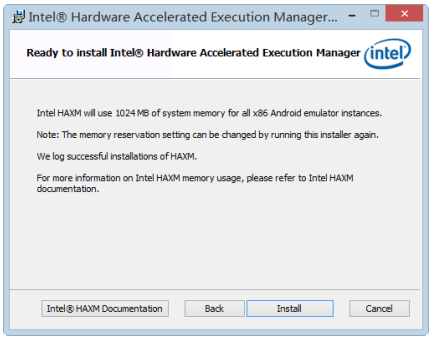


图 F-3　内存大小等提示信息界面

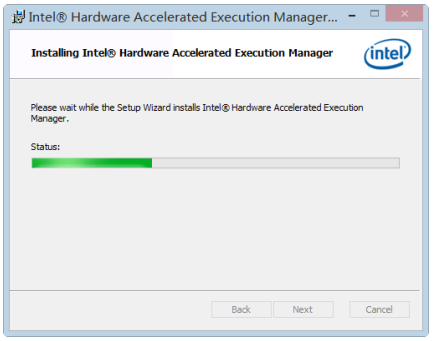


图 F-4　英特尔硬件加速执行管理器安装进度界面

3．测试英特尔硬件加速执行管理器是否安装成功

英特尔硬件加速执行管理器安装完成后即可开始使用，如果可运行，则可以使用英特尔虚拟化技术加速模拟。

安装完毕后，在命令行执行 sc query intelhaxm，如果 STATE 显示为“RUNNING”则表示安装成功，如图 F-6 所示。

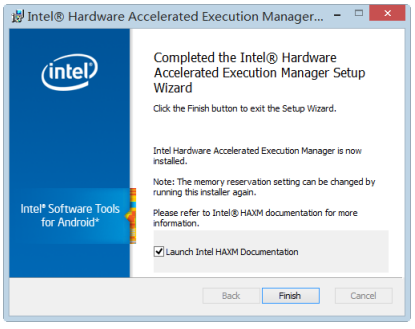


图 F-5　英特尔硬件加速执行管理器安装完成界面



图 F-6　测试英特尔硬件加速执行管理器是否安装成功

如果英特尔硬件加速执行管理器在运行，该命令将显示状态消息，状态“4”表明“运行中”。

【提示】：要停止英特尔硬件加速执行管理器，使用“sc stop intelhaxm”命令，要启动英特尔硬件加速执行管理器，使用“sc start intelhaxm”命令。

附录 G

下载、安装与配置模拟器 Genymotion

1．下载 Genymotion

（1）打开浏览器，输入网址 http://www.genymotion.net/，进入 Genymotion 的首页，如图 G-1 所示。



图 G-1　Genymotion 的首页

（2）Genymotion 并不能直接下载，需要注册账号并成功登录后才能下载。我们首先需要注册一个账号，这个账号要牢记，不仅下载时使用，以后创建虚拟设备时也需要使用。

在 Genymotion 的首页单击【注册】超链接，打开 Genymotion 登录页面，在该页面中单击【Create account】超链接即可打开注册页面进行注册操作，注册成功后登录，这里直接输入“用户名”和“密码”，如图 G-2 所示。

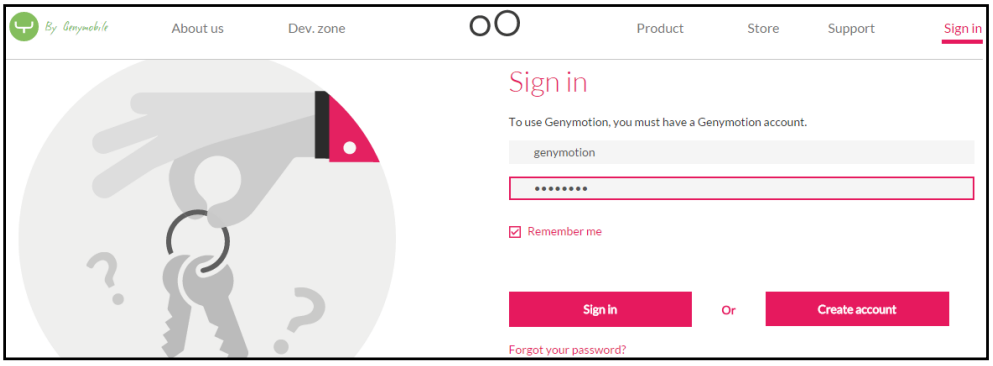


图 G-2　在 Genymotion 注册页面输入注册信息

输入正确用户名和密码后单击【Sign in】超链接即可到达下载页面，它分为两种情况，一种免费，一种付费，“Free”与“Business”两个版本的功能比较页面如图 G-3 所示。

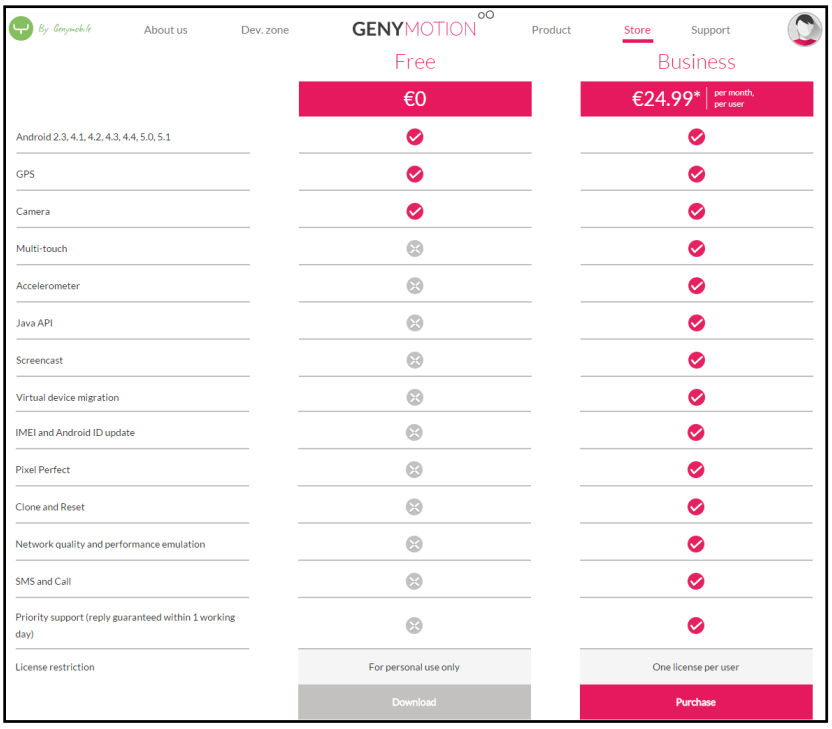


图 G-3　“Free”与“Business”两个版本的功能比较页面

这里我们下载 Free 版本，单击 Free 下面的【Download】按钮（虽然是灰色的，但是单击仍然有效)，就会进入 Genymotion 的下载界面，选择 Windows 版本，然后找到下载链接，如图 G-4 所示。

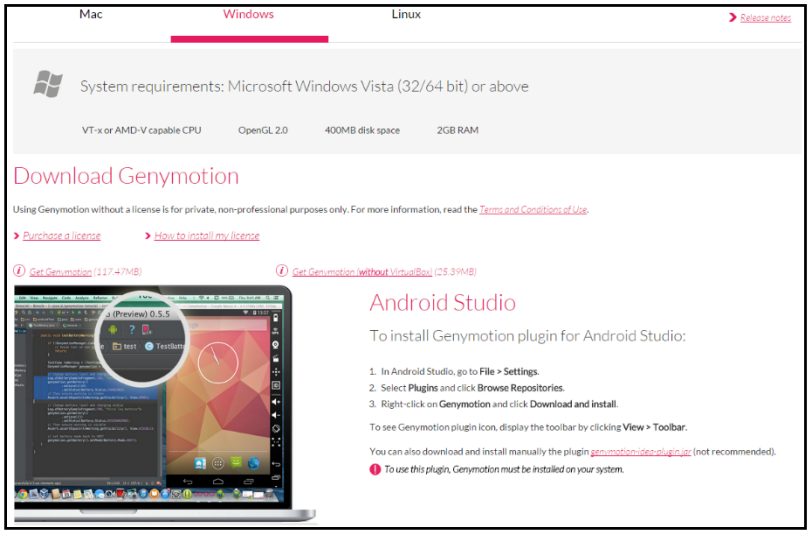


图 G-4　Genymotion 的下载界面

在该下载页面中单击【Get Genymotion】即可开始下载 Genymotion 的安装文件，单击右下角的【genymotion-idea-plugin.jar】即可下载插件。

Genymotion 下载完成，直接安装即可。

2．安装 Genymotion

双击 Genymotion 可执行的安装文件，例如“genymotion-2.4.0-vbox.exe”，启动 Genymotion 安装向导，首先显示欢迎界面，如图 G-5 所示。

单击【Next】按钮，进入“Select Destination Location”界面，设置 Genymotion 的安装文件夹，这里使用默认值，如图 G-6 所示。

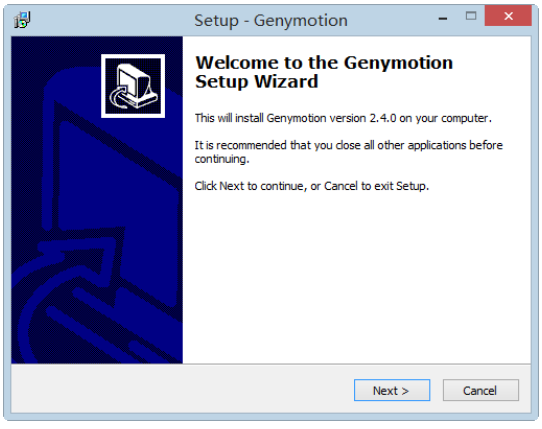


图 G-5　Genymotion 安装向导的“欢迎”界面

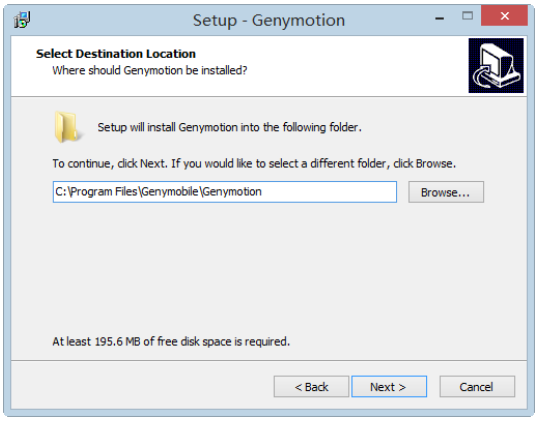


图 G-6　Genymotion 安装向导的“Select Destination Location”界面

单击【Next】按钮，进入“Select Start Menu Folder”界面，设置开始菜单位置和名称，这里使用默认值，如图 G-7 所示。

单击【Next】按钮，进入“Select Additional Tasks”界面，选中“Create a desktop icon”复选框，如图 G-8 所示。

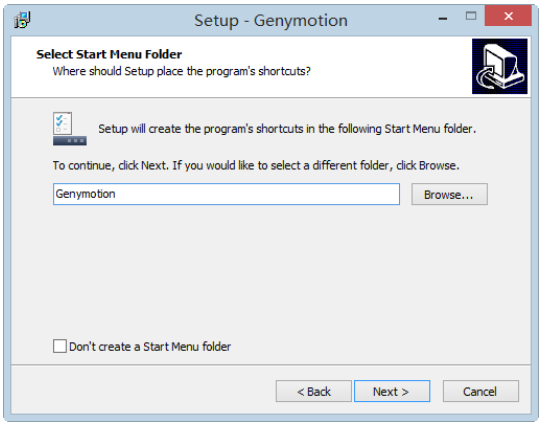


图 G-7　Genymotion 安装向导的“Select Start Menu Folder”界面

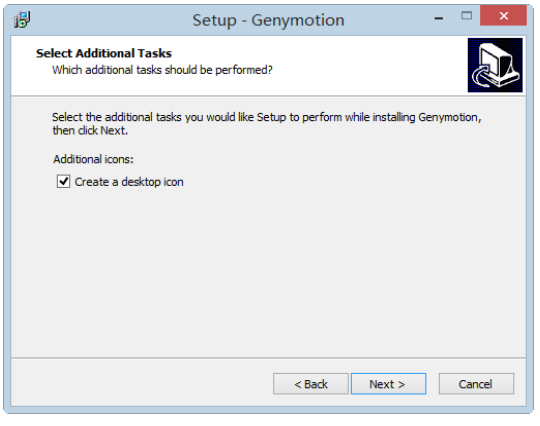


图 G-8　Genymotion 安装向导的“Select Additional Tasks”界面

单击【Next】按钮，进入“Ready to Install”界面，在该界面可以查看前几步的设置结果，如图 G-9 所示。

单击【Install】按钮，进入 Genymotion 的安装进度界面，如图 G-10 所示。

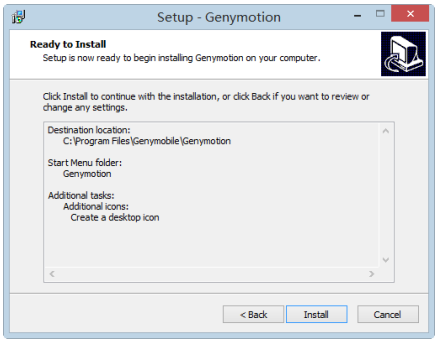


图 G-9　Genymotion 安装向导的“Ready to Install”界面

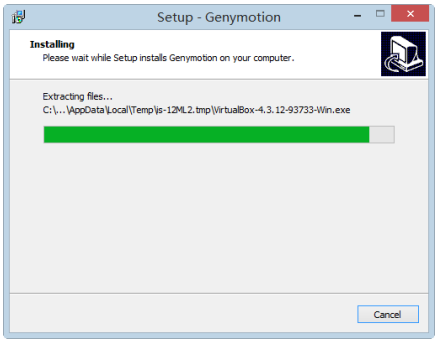


图 G-10　Genymotion 安装向导的“Installing”界面

Genymotion 安装完成后自动进入安装 VirtualBox 的启动界面，如图 G-11 所示。

单击【Next】按钮，进入 VirtualBox 安装向导的“Custom Setup”界面，选择需要安装的选项，如图 G-12 所示。



图 G-11　安装 VirtualBox 的启动界面

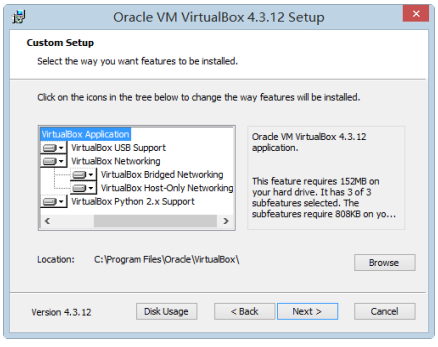


图 G-12　安装 VirtualBox 的“Custom Setup”界面

然后根据需要对“Create a shortcut on the desktop(创建桌面快捷方式)”、“Create a shortcut in the Quick Launch Bar(在快速启动栏创建快捷方式)”和“register file associations（寄存器文件关联）3 个复选框进行选择，这里 3 个复选框都选中，如图 G-13 所示。

单击【Next】按钮，出现“警告：网络接口”界面，如图 G-14 所示。

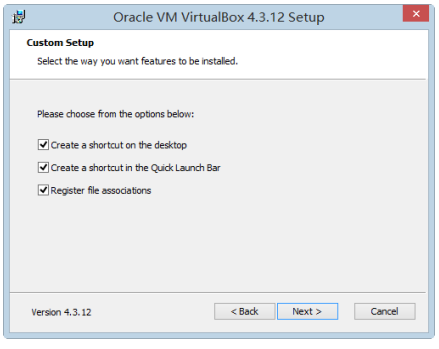


图 G-13　选择 3 个复选框



图 G-14　“警告：网络接口”界面

单击【Yes】按钮，出现“Ready to Install”界面，如图 G-15 所示。

单击【Install】按钮，开始安装，并出现 VirtualBox 的安装进度界面，如图 G-16 所示。

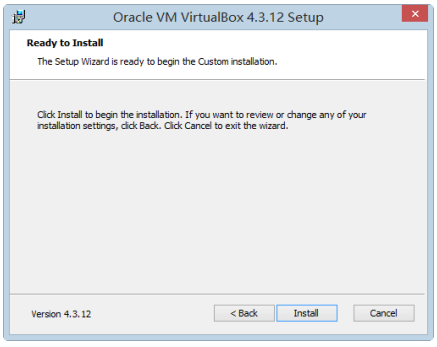


图 G-15　“Ready to Install”界面

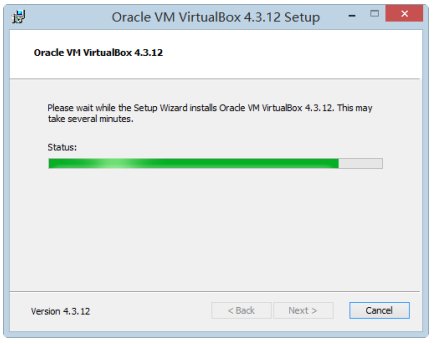


图 G-16　VirtualBox 的安装进度界面

VirtualBox 安装完成后，会弹出如图 G-17 所示的界面。

单击【Finish】按钮，会弹出如图 G-18 所示的 Genymotion 安装向导完成界面，单击【Finish】按钮退出安装。



图 G-17　VirtualBox 安装完成提示界面

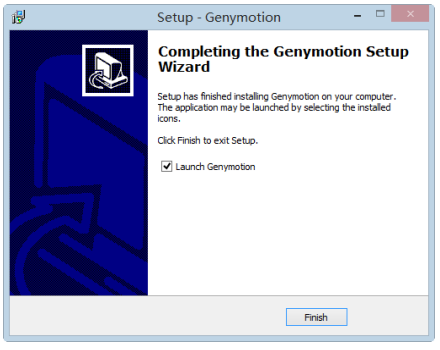


图 G-18　Genymotion 安装向导完成界面

3．Android 与 Genymotion 集成配置

Genymotion 安装完成后，接下来完成 Android Studio 集成 Genymotion 模拟器的操作。打开 Android studio，选择其“File”菜单中的“Settings”命令，打开“Settings”对话框，在该对话框左侧第 1 列中选择“Plugins”选项，如图 G-19 所示。

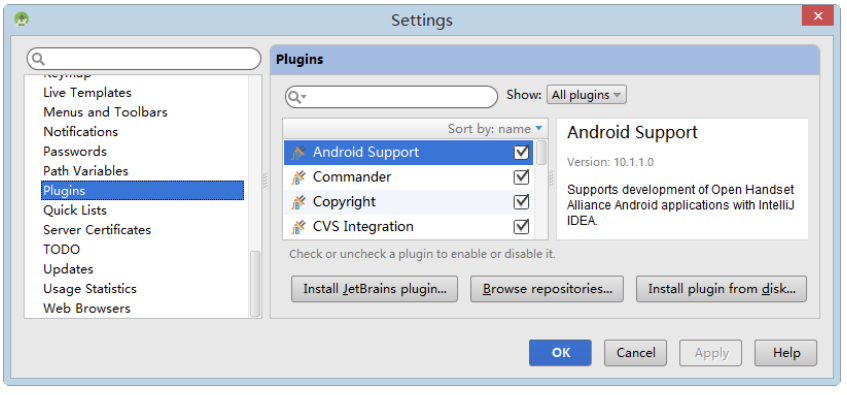


图 G-19　在“Settings”对话框中选择“Plugins”选项

然后单击【Install plugin from disk】按钮，在弹出的“Choose Plugin File”对话框中选择前面下载的插件 genymotion-idea-plugin.jar，单击【OK】按钮开始安装插件，如图 G-20 所示。安装完成弹出如图 G-21 所示“Plugins Changed”提示信息对话框，单击【Restart】按钮，重启 Android Studio。重启后，可以看到 Android Studio 的工具栏中多了一个按钮，如图 G-22 所示。

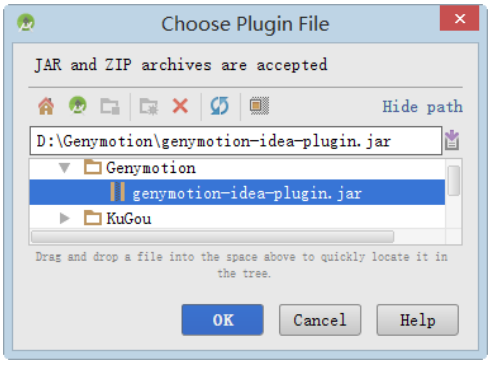


图 G-20　“Choose Plugin File”对话框

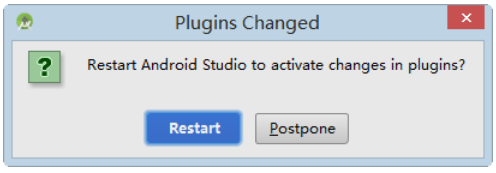


图 G-21　“Plugins Changed”提示信息对话框



图 G-22　插件 genymotion-idea-plugin.jar 安装完成后 Android Studio 的工具栏

Android Studio 重启后，重新打开“Settings”对话框，在“Settings”对话框中左侧第 1 列选择“Genymotion”选项，然后在右侧设置或输入 Genymotion 的安装路径，如图 G-23 所示。

4．添加 Android 模拟器

在桌面双击 Genymotion 快捷方式，启动 Genymotion，显示如图 G-24 所示的初始界面。

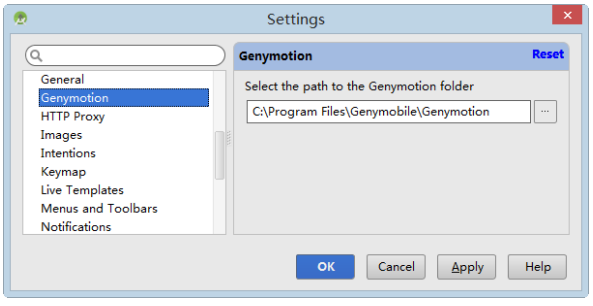


图 G-23　在“Settings”对话框中设置 Genymotion 的安装路径

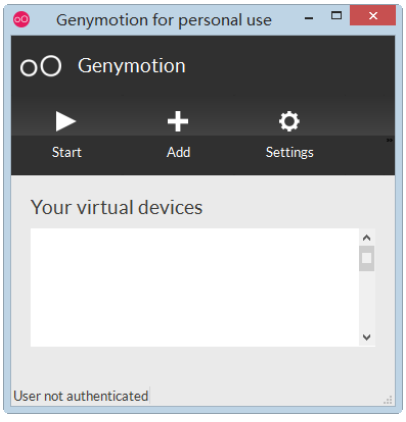


图 G-24　Genymotion 的初始界面

现在可以开始添加新的虚拟 Android 设备了，在 Genymotion 的工具栏中单击【Add】按钮，打开“Virtual device creation wizard”，在其“Select a new virtual device”界面，单击【Sign in】按钮，弹出【Authentication】对话框，在该对话框的“Username”文本框中输入已注册的用户名，在“Password”的密码框中输入正确的密码，如图 G-25 所示，然后单击【Sign in】按钮，开始登录。

登录成功后就会在“Available virtual devices（可用的虚拟设备）”列表中出现 Genymotion 支持的设备列表供选择下载，在“Android version”列表框中选择合适的 Android 版本，这里选择“5.0.0”，还可以在“Device model”列表选择模拟器模式，这里选择“All”，然后在模拟器机型列表中选择所需添加的模拟器，这里选择“Google Nexus 4-5.0.0-API21-768×1280”，如图 G-26 所示。

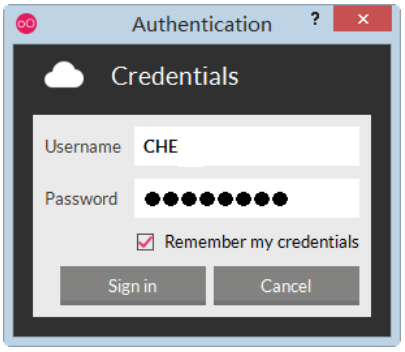


图 G-25　Genymotion 的登录界面

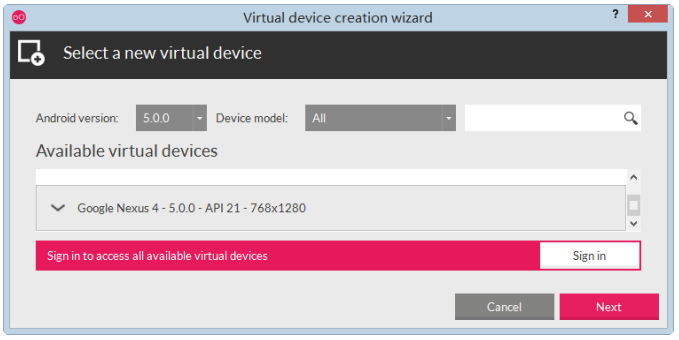


图 G-26　选择 Android 版本和模拟器机型

然后单击【Next】按钮，等待下载，下载完毕即可开始安装 Android 模拟器。成功添加了多个模拟器的 Genymotion 界面如图 G-27 所示，选择已添加的模拟器，就可以进行配置、删除等操作。

5．启动 Android 模拟器

在本机模拟器列表中选择已成功添加的模拟器，这里选择“Google Nexus 4-5.0.0-API21-768×1280”，然后在 Genymotion 工具栏中单击【Start】按钮，开始启动模拟器，启动画面如图 G-28 所示。

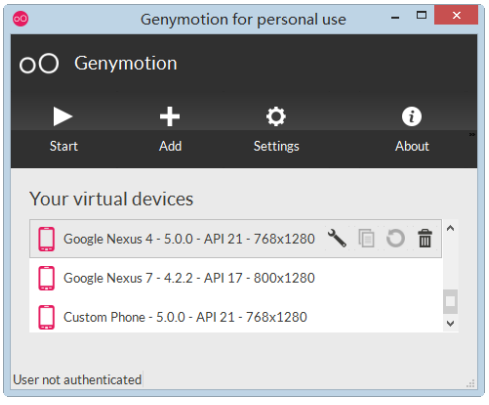


图 G-27　成功添加了多个模拟器的 Genymotion 界面

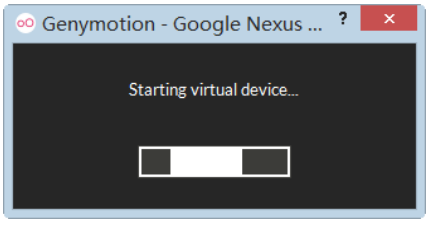


图 G-28　启动模拟器画面

模拟器启动成功后如图 G-29 所示，向右滑动锁，解锁后的模拟器界面如图 G-30 所示。

图 G-30 所示的这款模拟器支持电源管理、GPS 和摄像头，还添加了电源键，单击电源键就可以实现锁屏。减小音量键下面的那个按钮就是横屏和竖屏切换键，单击该键后可以在横屏和竖屏之间直接切换。单击中间的应用按钮，显示应用程序列表，如图 G-31 所示。



图 G-29　模拟器启动成功后的界面

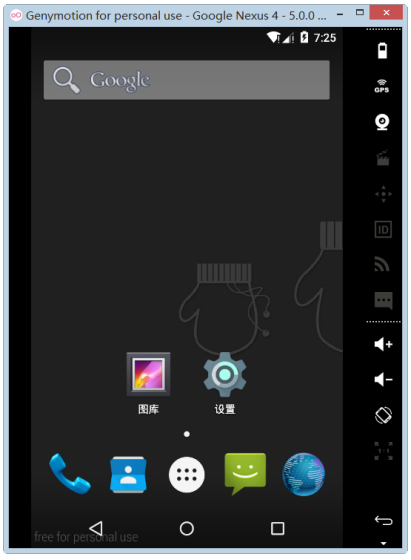


图 G-30　解锁后的模拟器界面



图 G-31　模拟器显示应用程序列表

【提示】：Android 原生就支持多国语言，只需要设置一下即可。图 G-31 所示的中文界面是经设置后的结果，首先单击列表中的【Settings】按钮进入功能设置界面，然后单击【Language & input】按钮，接着选择“中文（简体）”后，模拟器界面即变为中文了。

参 考 文 献

[1] 软件开发技术联盟．Android 开发实战．北京：清华大学出版社，2013
[2] 武永亮．Android 开发范例实战宝典．北京：清华大学出版社，2014
[3] 传智播客高教产品研发部．Android 移动应用基础教程．北京：中国铁道出版社，2015
[4] 倪红军，周巧扣．Android 开发工程师案例教程．北京：北京大学出版社，2014
[5] 余永佳，赵佩华．Android 应用开发基础．北京：机械工业出版社，2014
[6] 姚尚朗，靳岩．Android 开发入门与实战．北京：人民邮电出版社，2014
[7] 雷擎，伊凡．基于 Android 平台的移动互联网开发．北京：清华大学出版社，2014
[8] 陈长顺．Android 应用开发．北京：高等教育出版社，2014